```
TTTTTTTTTTTTTTT  TTTTTTTTTTTTTTT  DDDDDDDDDDDD     RRRRRRRRRRRR     VVV         VVV  RRRRRRRRRRRR
TTTTTTTTTTTTTTT  TTTTTTTTTTTTTTT  DDDDDDDDDDDDD    RRRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRRR
TTTTTTTTTTTTTTT  TTTTTTTTTTTTTTT  DDDDDDDDDDDDD    RRRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRRR
      TTT              TTT        DDD         DDD  RRR         RRR  VVV         VVV  RRR         RRR
      TTT              TTT        DDD         DDD  RRR         RRR  VVV         VVV  RRR         RRR
      TTT              TTT        DDD         DDD  RRR         RRR  VVV         VVV  RRR         RRR
      TTT              TTT        DDD         DDD  RRR         RRR  VVV         VVV  RRR         RRR
      TTT              TTT        DDD         DDD  RRR         RRR  VVV         VVV  RRR         RRR
      TTT              TTT        DDD         DDD  RRR         RRR  VVV         VVV  RRR         RRR
      TTT              TTT        DDD         DDD  RRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRR
      TTT              TTT        DDD         DDD  RRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRR
      TTT              TTT        DDD         DDD  RRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRR
      TTT              TTT        DDD         DDD  RRR   RRR       VVV         VVV  RRR   RRR
      TTT              TTT        DDD         DDD  RRR   RRR       VVV         VVV  RRR   RRR
      TTT              TTT        DDD         DDD  RRR   RRR       VVV         VVV  RRR   RRR
      TTT              TTT        DDD         DDD  RRR      RRR     VVV   VVV     RRR      RRR
      TTT              TTT        DDD         DDD  RRR      RRR     VVV   VVV     RRR      RRR
      TTT              TTT        DDD         DDD  RRR      RRR     VVV   VVV     RRR      RRR
      TTT              TTT        DDDDDDDDDDDD     RRR         RRR     VVV       RRR         RRR
      TTT              TTT        DDDDDDDDDDDDD    RRR         RRR     VVV       RRR         RRR
      TTT              TTT        DDDDDDDDDDDD     RRR         RRR     VVV       RRR         RRR
      TTT              TTT        DDDDDDDDDDDDD    RRR         RRR     VVV       RRR         RRR
      TTT              TTT        DDDDDDDDDDDD     RRR         RRR     VVV       RRR         RRR
```

```
Va
--
00
00
00
00
00
00
00
48
80
80
80
80
80
80
80
80
80
80
80
80
80
80
80
80
80
80
80
80
```

```
TTTTTTTTTT  TTTTTTTTTT  YY      YY    CCCCCCCC  HH      HH    AAAAAA    RRRRRRRR      OOOOOO
TTTTTTTTTT  TTTTTTTTTT  YY      YY    CCCCCCCC  HH      HH    AAAAAA    RRRRRRRR      OOOOOO
   TT          TT       YY      YY  CC          HH      HH  AA      AA  RR      RR  OO      OO
   TT          TT       YY      YY  CC          HH      HH  AA      AA  RR      RR  OO      OO
   TT          TT         YY  YY    CC          HH      HH  AA      AA  RR      RR  OO      OO
   TT          TT         YY  YY    CC          HH      HH  AA      AA  RR      RR  OO      OO
   TT          TT           YY      CC          HHHHHHHHHH  AA      AA  RRRRRRRR    OO      OO
   TT          TT           YY      CC          HHHHHHHHHH  AA      AA  RRRRRRRR    OO      OO
   TT          TT           YY      CC          HH      HH  AAAAAAAAAA  RR  RR      OO      OO
   TT          TT           YY      CC          HH      HH  AAAAAAAAAA  RR  RR      OO      OO
   TT          TT           YY      CC          HH      HH  AA      AA  RR    RR    OO      OO      ....
   TT          TT           YY      CC          HH      HH  AA      AA  RR    RR    OO      OO      ....
   TT          TT           YY        CCCCCCCC  HH      HH  AA      AA  RR      RR    OOOOOO        ....
   TT          TT           YY        CCCCCCCC  HH      HH  AA      AA  RR      RR    OOOOOO        ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

Table of contents

(2) 237 Declarations
(3) 264 TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
(4) 338 WRITING - WRITE state action routine
(5) 415 BURST/STRING - SET UP OUTPUT FOR BURST MODE
(6) 456 FORMAT - FORMAT STRING FOR OUTPUT
(7) 534 OUTPUTANDWAIT - OUTPUT CHARACTER AND WAIT FOR INTERRUPT
(8) 568 FORMAT_CHAR - FORMAT CHARACTER FOR OUTPUT
(9) 621 ADJUST_CURSOR - Increment cursor and set wrap if needed
(10) 651 FORMAT_LOCAL - FORMAT CHARACTER FOR OUTPUT
(11) 697 CHARACTER OUTPUT FORMAT ACTION ROUTINES
(11) 698 BACKSPACE - output a backspace
(12) 724 CARRIAGE - format a carriage return
(13) 803 CTRLZ - output control-Z
(14) 838 ESCAPE - format an escape character
(15) 902 LINEFEED - format a line feed
(16) 950 TAB - Output a tab.
(17) 1012 VTAB, FORM - output a vertical tab, or form feed
(18) 1059 DISPATCHER SERVICE ROUTINES
(19) 1061 BACKSPACING - OUTPUT N BACKSPACES
(20) 1082 CURSOROVRFLOW - insert newline to handle end of line
(21) 1119 EOLSEEN - handle end of line condition
(22) 1159 FILLING - continue outputting fill characters
(23) 1188 MULTIECHOING - Continue outputting multiecho sequence
(24) 1230 SENDLINEFEED - output a line feed
(25) 1255 ESCAPE SEQUENCE PARSING SERVICES
(26) 1273 ESCSYNTAX -- CHECK ESCAPE SEQUENCE SYNTAX
(27) 1347 MOVEREADATA - MOVE CHARACTER FROM TYPEAHEAD TO READ BUFFER
(28) 1481 MOVEREADATA SERVICE ROUTINES
(37) 1736 Specail input character dispatcher
(39) 1812 Post typeahead character action routines
(56) 2335 EDITREAD - READ EDITING STATE
(67) 2542 Read service routines
(75) 2934 Read passall
(76) 3042 READ WITH VERIFICATION
(79) 3434 READ VERIFY ESCAPE TERMINATOR ROUTINE
(80) 3475 End of module

```
0000     1          .TITLE  TTYCHARO - Terminal driver character output routine
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ; Facility:
0000    29 ;
0000    30 ;       VAX/VMS TERMINAL DRIVER
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       THIS MODULE CONTAINS ROUTINES USED FOR THE OUTPUT OF CHARACTERS.
0000    35 ;
0000    36 ;       ***** This module never destroys the contents of R0 *****
0000    37 ;
0000    38 ; AUTHOR:
0000    39 ;
0000    40 ;       R.HEINEN 11-AUG-1976
0000    41 ;
0000    42 ; Revision history:
0000    43 ;
0000    44 ;       V03-046 MIR01424        Michael I. Rosenblum            16-Jun-1984
0000    45 ;               Fix bug with read with 0 timeout that creates the typeahead
0000    46 ;               buffer, reported in qar 1424.
0000    47 ;
0000    48 ;       V03-045 MIR0450         Michael I. Rosenblum            27-JUN-1984
0000    49 ;               Make the port bit pc_nocrlf set the state skiplf
0000    50 ;               Fix bug in escape introducer code that caused it not to
0000    51 ;               recognize csi.
0000    52 ;
0000    53 ;       V03-044 MIR0400         Michael I. Rosenblum            10-Apr-1984
0000    54 ;               Filter null characters from escape sequences.
0000    55 ;
0000    56 ;       V03-043 MIR0390         Michael I. Rosenblum            04-Apr-1984
0000    57 ;               Make the exit string echo thru the tables vector.
```

```
0000     58 ;
0000     59 ;       V03-042 MIR0370         Michael I. Rosenblum            20-Mar-1984
0000     60 ;               Fix problems with tabs to the left of the curent position,
0000     61 ;               Lines that wrap in either the prompt or initial string.
0000     62 ;               Add port control word modifier to stop free linefeeds
0000     63 ;               After typed <CR>.
0000     64 ;
0000     65 ;       V03-042 MIR0310         Michael I. Rosenblum            07-Feb-1984
0000     66 ;               Fix bugs.
0000     67 ;               Noformat reads shouldn't wrap.
0000     68 ;               Ignore SKIPCRLF and EDITREAD if we are wrapping.
0000     69 ;               Don't change esc to $ if EDITREAD is set.
0000     70 ;               Move to beginning of line should rewrite the whole line
0000     71 ;                 if the origional cursor position was non-zero.
0000     72 ;               Read nofltr should obey uppercase rules.
0000     73 ;               Passall should not imply termnoecho.
0000     74 ;               Fix delete in left justified read verify noecho fields
0000     75 ;                 to not echo.
0000     76 ;
0000     77 ;       V03-041 MIR0300         Michael I. Rosenblum            30-Jan-1984
0000     78 ;               Make the uparrow key be command recall.
0000     79 ;
0000     80 ;       V03-040 MIR0100         Michael I. Rosenblum            24-Oct-1983
0000     81 ;               Fix bug that would cause characters to be echoed in no-echo
0000     82 ;               mode.
0000     83 ;
0000     84 ;       V03-039 MIR0084         Michael I. Rosenblum            25-Aug-1983
0000     85 ;               Fix problem with terminating a right justified numeric
0000     86 ;               auto-tab field in read verify.
0000     87 ;
0000     88 ;       V03-038 MIR0082         Michael I. Rosenblum            02-Aug-1983
0000     89 ;               Fix bug in recall logic that did not force a return before
0000     90 ;               the recall was done.  Fix bug in delete after wrap logic.
0000     91 ;
0000     92 ;       V03-037 MIR0081         Michael I. Rosenblum            01-Aug-1983
0000     93 ;               Remove $brddef call (obsoleted).
0000     94 ;
0000     95 ;       V03-036 MIR0080         Michael I. Rosenblum            28-Jul-1983
0000     96 ;               Reposition code in the module.
0000     97 ;
0000     98 ;       V03-035 MIR0070         Michael I. Rosenblum            13-Jul-1983
0000     99 ;               Fix bug that would cause an escape sequence at the beginning
0000    100 ;               of the line to return invalid data if deleted.
0000    101 ;               Make check in MOVEREADATA for buffer full condition at
0000    102 ;               the top of the loop to make sure that the users buffer
0000    103 ;               can not be overfilled if the initial string fills the
0000    104 ;               data buffer.
0000    105 ;
0000    106 ;       V03-034 MIR0053         Michael I. Rosenblum            27-Jun-1983
0000    107 ;               Fix bug in ESCAPE_TAB where it would pass the address
0000    108 ;               into the ESCINIT routine rather than the character
0000    109 ;
0000    110 ;       V03-033 MIR0051         Michael I. Rosenblum            23-Jun-1983
0000    111 ;               Fix bug caused when escape sequences are UNUSED.
0000    112 ;               Cause simceol to exit to movecursor on hard-copy terminals.
0000    113 ;               Make routines that checked CSI and ESC call a common routine.
0000    114 ;               Fix bug that would cause writes on non-hardcopy terminals
```

```
0000   115 ;                To use the single character code path following the wrap point.
0000   116 ;
0000   117 ;        V03-032 MIR0053         Michael I. Rosenblum            10-May-1983
0000   118 ;                Fix bug in clear to end of line emulation.
0000   119 ;
0000   120 ;        V03-031 MIR0052         Michael I. Rosenblum            27-May-1983
0000   121 ;                Add delete character to read verify, make read verify
0000   122 ;                accept fill characters in the initial string and as
0000   123 ;                data and display the clear character.
0000   124 ;                Add software emulation of clear to end of line for non-
0000   125 ;                ansi terminals.
0000   126 ;
0000   127 ;        V03-030 MIR0050         Michael I. Rosenblum            11-May-1983
0000   128 ;                Remove code that specail cased broadcasts.
0000   129 ;
0000   130 ;        V03-029 MIR0049         Michael I. Rosenblum            06-May-1983
0000   131 ;                Add code to handle wrapping correctly.
0000   132 ;
0000   133 ;        V03-028 MIR0041         Michael I. Rosenblum            29-Apr-1983
0000   134 ;                Fix bug in the new passall code path
0000   135 ;
0000   136 ;        V03-027 MIR0030         Michael I. Rosenblum            30-Mar-1983
0000   137 ;                Integrate TTRDVFY with the terminal driver, code included
0000   138 ;                in this module
0000   139 ;
0000   140 ;        V03-026 MIR0029         Michael I. Rosenblum            22-Mar-1983
0000   141 ;                Add code for insert/overstrike mode.
0000   142 ;
0000   143 ;        V03-025 RKS0025         RICK SPITZ                      19-MAR-1983
0000   144 ;                Change references of UCB$L_TT_DEVDP1 TO UCB$L_DEVDEPND2
0000   145 ;
0000   146 ;        V03-024 MIR6026         Michael I. Rosenblum            11-Mar-1983
0000   147 ;                Change recall to not terminate a small read when a recall
0000   148 ;                is done from a long recall buffer.
0000   149 ;
0000   150 ;        V03-023 MIR2026         Michael I. Rosenblum            07-Mar-1983
0000   151 ;                Fix bug in EDITREAD prompt echoing when there is no prompt
0000   152 ;                to echo.
0000   153 ;
0000   154 ;        V03-022 MIR1026         Michael I. Rosenblum            1-Mar-1983
0000   155 ;                Add $TRMDEF to the list of required symbol definitions
0000   156 ;
0000   157 ;        V03-021 MIR0026         Michael I. Rosenblum            10-Feb-1983
0000   158 ;                Add code to impliment input line editing.
0000   159 ;
0000   160 ;        V03-020 MIR0025         Michael I. Rosenblum            18-Feb-1983
0000   161 ;                Fix bug in carriage control logic that caused a carriage return
0000   162 ;                in a prompted read to get a free linefeed.
0000   163 ;
0000   164 ;        V03-019 MIR0024         Michael I. Rosenblum            28-Jan-1983
0000   165 ;                Change code to reflect redefinition of the read packet.
0000   166 ;
0000   167 ;        V03-018 MIR0017         Michael I. Rosenblum            05-Jan-1983
0000   168 ;                Add powerfail bit to the unit state vector table to remove
0000   169 ;                powerfail check for each character.  Restructure powerfail
0000   170 ;                code to fall into a common code path in TTYSUB.  Change
0000   171 ;                return status of TTY$GETNEXTCHAR to include a byte value
```

```
0000   172 ;                in the UCB, this will move the information from the condition
0000   173 ;                code bits.
0000   174 ;
0000   175 ;        V03-017 MIR0016         Michael I. Rosenblum            29-Dec-1982
0000   176 ;                Replace time calculation code with TIMSET macro call
0000   177 ;                This change will allow us to change the time calculation
0000   178 ;                algorhythm globaly if necessary, and will also allow
0000   179 ;                us to globaly turn on and off the time calculations.
0000   180 ;
0000   181 ;        V03-016 MIR0015         Michael I. Rosenblum            22-Dec-1982
0000   182 ;                Change all calls to the terminal port drivers to refer
0000   183 ;                to the class jacket routines. Remove DMA code from the
0000   184 ;                class driver.
0000   185 ;
0000   186 ;        V03-015 MIR0013         Michael I. Rosenblum            16-Dec-1982
0000   187 ;                Fix up refferences to new ucb structure
0000   188 ;
0000   189 ;        V03-014 MIR0011         Michael I. Rosenblum            18-Nov-1982
0000   190 ;                Change MULTIECHO to take a count in UCB$W_TT_MULTILEN and
0000   191 ;                an address in UCB$L_TT_MULTI, illiminate the code for
0000   192 ;                zero terminated multiecho strings.
0000   193 ;                Change STRTMULTI to take a counted string and convert
0000   194 ;                it into a length and address.
0000   195 ;                Add STRTMULTI_1 to take the place of STRTMULTI.
0000   196 ;                Add code to make Control-R and Control-U look prety on
0000   197 ;                ANSI crt terminals.
0000   198 ;                Change CONTROLR to EDITREAD also change the internals of
0000   199 ;                CONTROLR to use MULTIECHO.
0000   200 ;                Change the meaning of the CTRLR bit to indicate that the
0000   201 ;                prompt and data string from a read buffer are being clocked
0000   202 ;                out by multiecho, this is the only use for CTRLR.
0000   203 ;                Add SKIPCRLF to indicate that a linefeed following a <CR>
0000   204 ;                in the beginning of the prompt string is to be skiped.
0000   205 ;                Remove HOLDSCREEN code.
0000   206 ;
0000   207 ;        V03-013 MIR0012         Michael I. Rosenblum    19-Nov-1982
0000   208 ;                Fix bug that caused exception when a read was posted
0000   209 ;                that caused a create of the typeahead buffer.
0000   210 ;
0000   211 ;        V03-012 MIR0010         Michael I. Rosenblum    09-Nov-1982
0000   212 ;                Move the address of the terminator mask, and the length
0000   213 ;                of the prompt string from the IRP into the terminal read
0000   214 ;                packet.  Also move the count of the characters in the
0000   215 ;                buffer from the UCB into the terminal typeahead buffer packet.
0000   216 ;                Make backspace look like delete (without beeing delete)
0000   217 ;
0000   218 ;        V03-011 RKS0011         RICK SPITZ                      19-OCT-1982
0000   219 ;                INSURE THAT R5 IS PRESERVED WHEN DMA FORK TAKEN
0000   220 ;
0000   221 ;        V03-009 RKS0010         RICK SPITZ                      23-SEP-1982
0000   222 ;
0000   223 ;                CHANGE LOGIC IN FORMAT TO INSURE THAT HARDCOPY TERMINALS
0000   224 ;                SET TO NO WRAP DO NOT OUTPUT DATA BEYOND LINE WIDTH
0000   225 ;
0000   226 ;                CORRECT PROBLEM WITH XON LOGIC TO INSURE THAT OVERFLOW
0000   227 ;                ERRORS ARE ALWAYS REPORTED
0000   228 ;
```

```
0000    229 ;       V03-008 MIR0003         Michael I. Rosenblum    12-Aug-1982
0000    230 ;               Fix bug where multi escape sequences in a TTRDVFY prompt
0000    231 ;               Would not work.
0000    232 ;
0000    233 ;       V03-007 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    234 ;               Added $SSDEF.
0000    235 ;--
```

```
                        .SBTTL  Declarations

;
; EXTERNAL SYMBOLS
;

                        $CRBDEF                         ; DEFINE CRB
                        $IODEF                          ; DEFINE I/O FUNCTION CODES
                        $IPLDEF                         ; DEFINE IPL'S
                        $IRPDEF                         ; DEFINE IRP
                        $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                        $PRIDEF                         ; DEFINE PRIORITY CLASSES
                        $RSNDEF                         ; DEFINE RESOURCE NUMBERS
                        $SSDEF                          ; DEFINE SYSTEM STATUS CODES
                        $UBADEF                         ; DEFINE UBA OFFSETS
                        $UCBDEF                         ; DEFINE UCB
                        $TRMDEF                         ; TERMINAL ITEMLIST BIT DEFINITIONS
                        $TTYDEF                         ; DEFINE TERMINAL DRIVER SYMBOLS
                        $TTDEF                          ; DEFINE TERMINAL CHARACTERISTICS
                        $TT2DEF                         ; DEFINE EXTENDED TERMINAL CHARS
                        $VECDEF                         ; DEFINE CRB VECTOR
                        $TTYMACS                        ; DEFINE TERMINAL MACROS
                        $TTYDEFS                        ; DEFINE TERMINAL DEFINITIONS

                        .PSECT  $$$115_DRIVER,LONG      ; DEFINE NON-PAGED PSECT
```

```
                     0000   264          .SBTTL  TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
                     0000   265 ;++
                     0000   266 ; TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
                     0000   267 ;
                     0000   268 ; FUNCTIONAL DESCRIPTION:
                     0000   269 ;
                     0000   270 ; THIS ROUTINE RETURNS WITH THE NEXT CHARACTER(S) TO BE OUTPUT ON THE UNIT.
                     0000   271 ;
                     0000   272 ; INPUTS:
                     0000   273 ;
                     0000   274 ;       R5 = UCB ADDRESS
                     0000   275 ;
                     0000   276 ; OUTPUTS:
                     0000   277 ;
                     0000   278 ;       R0 = ALWAYS PRESERVED!!
                     0000   279 ;
                     0000   280 ;       R3 =    0       AND CC = ZERO - NO CHARACTER TO OUTPUT
                     0000   281 ;               CHAR    AND CC = PLUS - SINGLE CHARACTER TO OUTPUT
                     0000   282 ;               ADDRESS AND CC = NEG  - BURST (R2 = LENGTH)
                     0000   283 ;                                       (ADDRESS AND LENGTH ALSO IN UCB)
                     0000   284 ;
                     0000   285 ;       R5 = UCB ADDRESS
                     0000   286 ;       UCB$B_TT_OUTYPE = -1 BIRST
                     0000   287 ;                                  ADDRESS IN R3 AND UCB$L_TT_OUTADR
                     0000   288 ;                                  LENGTH IN R2  AND UCB$W_TT_OUTLEN
                     0000   289 ;                               0 NO CHARACTER TO OUTPUT
                     0000   290 ;                               1 SINGLE CHARACTER TO OUTPUT IN R3
                     0000   291 ;--
                     0000   292
                     0000   293 TTY$GETNEXTCHAR::
          52  00B8 C5  9E  0000   294          MOVAB   UCB$Q_TT_STATE(R5),R2   ; ADDRESS STATE OF UNIT
                     0005   295 ;
                     0005   296 ; GET CURRENT STATE OF THE OUTPUT
                     0005   297 ;
                     0005   298 GETNEXTCHAR:                                  ; INTERNAL ENTRY
      62  00000080 8F  D1  0005   299          CMPL    #TTY$M_ST_WRITE,(R2)    ; TRY WRITING GENERAL CASE FIRST
                  3F  13  000C   300          BEQL    WRITING                 ; IF EQL THEN GO
      62  00001000 8F  D1  000E   301          CMPL    #TTY$M_ST_READ,(R2)     ; TRY READING GENERAL CASE NEXT
                  03  12  0015   302          BNEQ    10$                     ; NO, GO TO 10$
                050C  31  0017   303          BRW     MOVEREADATA             ; IF EQL THEN GO
                     001A   304 10$:
    54  62  0D  00  EA  001A   305          FFS     #0,#TTY$V_ST_READ+1,(R2),R4; GET THE NEXT MOST IMPORTANT OUTPUT STAT
                     001F   306          CASE    R4,TYPE=B,<-                    ; ENTER THAT STATE'S ROUTINE
                     001F   307          POWERREST,-                     ; go to the power fail code
                     001F   308          INTEXIT,-                       ; CONTROL S
                     001F   309          FILLING,-                       ; FILL IN PROGESS
                     001F   310          CURSOROVRFLOW,-                 ; CURSOR OVERFLOW
                     001F   311          SENDLINEFEED,-                  ; LINE FEED TO BE INSERTED
                     001F   312          BACKSPACING,-                   ; BACKING UP IN PROGRESS
                     001F   313          MULTIECHOING,-                  ; MULTI ECHO IS PROGRESS
                     001F   314          WRITING,-                       ; WRITE IN PROGRESS
                     001F   315          EOLSEEN,-                       ; EOL SEEN
                     001F   316          EDITREAD,-                      ; READ EDITING OUTPUT STATE
                     001F   317          RDVERIFY,-                      ; GO TO READ VERIFICATION
                     001F   318          RECALLING,-                     ; RECALLING THE LAST COMMAND
                     001F   319          MOVEREADATA>                    ; READ IN PROGRESS
                     003D   320 ;
```

```
                     003D   321 ; EXIT INTERRUPT
                     003D   322 ;
                     003D   323 INTEXIT:                                     ; EXIT INTERRUPT
010B C5    94        003D   324          CLRB    UCB$B_TT_OUTYPE(R5)        ; SET NO RETURN CHARACTER
           05        0041   325          RSB                                ; RETURN
                     0042   326 ;
                     0042   327 ; POWERFAIL DETECTED
                     0042   328 ;
                     0042   329 POWERREST:
   FFBB'   30        0042   330          BSBW    RESTART                    ; CALL THE POWERFAIL RESTART CODE THEN
   FFBD    31        0045   331          BRW     GETNEXTCHAR                ; CONTINUE AGAIN
                     0048   332
                     0048   333
                     0048   334 INTEXIT1:                                    ; EXIT AND RESTORE UCB ADDRESS
     55 8ED0         0048   335          POPL    R5
     F0   11         004B   336          BRB     INTEXIT
```

```
                        004D    338          .SBTTL  WRITING - WRITE state action routine
                        004D    339
                        004D    340 ;++
                        004D    341 ; WRITING - CONTINUE WRITE I/O OUTPUT
                        004D    342 ;
                        004D    343 ; FUNCTIONAL DESCRIPTION:
                        004D    344 ;
                        004D    345 ; THIS ROUTINE IS ENTERED WHEN ALL EXTRA OUTPUT IS COMPLETE
                        004D    346 ; AND A WRITE OPERATION IS IN PROGRESS.
                        004D    347 ;
                        004D    348 ; THE NEXT AVAILABLE CHARACTER IS EXTRACTED FROM THE USER BUFFER
                        004D    349 ; AND GIVEN TO THE ECHO FORMAT ROUTINES TO OUTPUT CORRECTLY.
                        004D    350 ;
                        004D    351 ; IF CONTROL O HAS STOPPED THE OUTPUT THE OPERATION IS COMPLETED.
                        004D    352 ;
                        004D    353 ; INPUTS:
                        004D    354 ;
                        004D    355 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        004D    356 ;       R5 = UCB ADDRESS
                        004D    357 ;
                        004D    358 ; Implicit inputs:
                        004D    359 ;
                        004D    360 ;       UCB$L_TT_WRTBUF - address of write buffer
                        004D    361 ;
                        004D    362 ; OUTPUTS:
                        004D    363 ;
                        004D    364 ;       R2 = ADDRESS OF THE TERMINAL STATE VECTOR
                        004D    365 ;       R3 = CHARACTER TO OUTPUT IF ANY
                        004D    366 ;       R5 = UCB ADDRESS
                        004D    367 ;--
                        004D    368
                        004D    369          .ENABLE LSB
                        004D    370 WRITING:
54   00D4 C5    D0      004D    371          MOVL    UCB$L_TT_WRTBUF(R5),R4  ; Get address of buffer block.
                        0052    372          IF_STATE CTRLO,60$              ; COMPLETE I/O OF CONTROL O
  53   1C A4    D0      0056    373          MOVL    TTY$L_WB_NEXT(R4),R3    ; GET ADDRESS OF NEXT CHARACTER
  20 A4   53    D1      005A    374          CMPL    R3,TTY$L_WB_END(R4)     ; DONE?
         09     1E      005E    375          BGEQU   50$                     ; IF GEQU THEN YES
                        0060    376          IF_STATE <PASALL,WRTALL>,BURST
       006D     31      0066    377          BRW     FORMAT                  ; FORMAT FOR OUTPUT
                        0069    378
                        0069    379 ;
                        0069    380 ; WRITE I/O DONE
                        0069    381 ;
                        0069    382
                        0069    383 50$:                                    ; Normal return.
                        0069    384          IF_NOT_STATE    <PASALL,WRTALL>,55$
    00FC C5    B4      006F    385          CLRW    UCB$W_TT_CURSOR(R5)     ; AT CONCLUSION OF PASSALL/WRITALL
    00FF C5    94      0073    386          CLRB    UCB$B_TT_LASTC(R5)      ; STATE IS UNKNOWN
                        0077    387 55$:
         01    B0      0077    388          MOVW    #SS$_NORMAL,-           ; Load success status code into
   28 A4               0079    389                  TTY$W_WB_STATUS(R4)     ; buffer header.
         06    11      007B    390          BRB     70$                     ; CONTINUE
                        007D    391
                        007D    392 ;
                        007D    393 ; CONTROL O TYPED OR CONTROL Y OR C
                        007D    394 ;
```

```
                        007D   395
                        007D   396 60$:                                    ; Control-O.
            0609 8F     B0  007D   397          MOVW    #SS$_CONTROLO,-     ; Load control-O status code
              28 A4         0081   398                  TTY$W_WB_STATUS(R4) ; into buffer header.
                        0083   399
                        0083   400 70$:                                    ; Complete transfer.
              1C A4     C3  0083   401          SUBL3   TTY$L_WB_NEXT(R4),- ; Calculate number of characters
           53   20 A4         0086   402                  TTY$L_WB_END(R4),R3 ; not output to terminal.
           2A A4    53   A2  0089   403          SUBW    R3,TTY$W_WB_BCNT(R4) ; Adjust transfer count.
                FF70'    30  008D   404          BSBW    TT$WRITEDONE        ; Do I/O done.
                FF6D     31  0090   405          BRW     TTY$GETNEXTCHAR     ; Get the next character.
                        0093   406
                        0093   407
                        0093   408 WRITE_END:
                        0093   409          IF_STATE CTRLO,60$                 ; CONTROL O
                  EA    11  0097   410          BRB     70$                 ; COMPLETE WITH CURRENT STATUS
                        0099   411
                        0099   412          .DISABLE        LSB
                        0099   413
```

```
                     0099   415          .SBTTL BURST/STRING - SET UP OUTPUT FOR BURST MODE
                     0099   416 ;++
                     0099   417 ; BURST/STRING -- SET UP OUTPUT FOR BURST MODE
                     0099   418 ;
                     0099   419 ; FUNCTIONAL DESCRIPTION:
                     0099   420 ;
                     0099   421 ; THIS ROUTINE RETURNS THE ADDRESS AND LENGTH OF THE CURRENT
                     0099   422 ; OUTPUT TO THE CALLER. DUETIM IS COMPUTED BASED ON THE LENGTH OF THE
                     0099   423 ; OUTPUT. THE CURRENT WRITE POINTERS ARE UPDATED TO REFLECT THE
                     0099   424 ; NUMBER OF CHARACTERS INCLUDED IN THE BURST.
                     0099   425 ;
                     0099   426 ; INPUTS:
                     0099   427 ;       R3 = ADDRESS OF NEXT CHARACTER
                     0099   428 ;       R4 = TWP ADDRESS
                     0099   429 ;       R5 = UCB ADDRESS
                     0099   430 ;
                     0099   431 ; OUTPUTS:
                     0099   432 ;
                     0099   433 ;       R3 = ADDRESS OF NEXT CHARACTER
                     0099   434 ;       R5 = UCB ADDRESS
                     0099   435 ;       UCB$L_TT_OUTADR = ADDRESS OF START OF BURST
                     0099   436 ;       UCB$L_TT_OUTLEN = LENGHT OF BURST
                     0099   437 ;--
                     0099   438 BURST:
 52   20 A4   53   C3 0099   439          SUBL3   R3,TTY$L_WB_END(R4),R2          ; COMPUTE LENGTH OF RECORD
                     009E   440
                     009E   441 STR_EXIT:
      52   52   3C   009E   442          MOVZWL  R2,R2                           ; CONVERT TO WORD VALUE
    1C A4   52   C0   00A1   443          ADDL    R2,TTY$L_WB_NEXT(R4)            ; UPDATE NEXT CHARACTER ADDRESS
  0120 C5   52   B0   00A5   444          MOVW    R2,UCB$W_TT_OUTLEN(R5)          ; SET OUTPUT SIZE
  011C C5   53   D0   00AA   445          MOVL    R3,UCB$L_TT_OUTADR(R5)          ; SET OUTPUT ADDRESS
                     00AF   446
                     00AF   447 STR_TIMESET:
                     00AF   448          TIMSET  R2,R1,LOCKOUTPUT                ; COMPUTE THE TIME PER CHARACTER
                     00D0   449                                                  ; AND SETUP INTERUPTS
010B C5   01   8E    00D0   450          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; SET NEGATIVE CC TO SIGNAL
                     00D5   451                                                  ; STRING OUTPUT
               05    00D5   452          RSB
                     00D6   453
                     00D6   454
```

```
                     00D6   456          .SBTTL FORMAT - FORMAT STRING FOR OUTPUT
                     00D6   457 ;++
                     00D6   458 ; FORMAT -- FORMAT STRING FOR OUTPUT
                     00D6   459 ; FUNCTIONAL DESCRIPTION:
                     00D6   460 ;
                     00D6   461 ; THIS ROUTINE RETURNS THE ADDRESS AND LENGTH OF A PORTION OF THE
                     00D6   462 ; CURRENT FORMATTED OUTPUT STRING TO THE CALLER.
                     00D6   463 ; IF NO SIGNIFICANT PORTION OF THE STRING CAN BE FOUND, IT
                     00D6   464 ; DEFAULTS TO SINGLE CHARACTER FORMAT MODE.
                     00D6   465 ; DUETIM IS COMPUTED BASED ON THE LENGTH OF THE
                     00D6   466 ; OUTPUT. THE CURRENT WRITE POINTERS ARE UPDATED TO REFLECT THE
                     00D6   467 ; NUMBER OF CHARACTERS INCLUDED IN THE BURST.
                     00D6   468 ;
                     00D6   469 ;
                     00D6   470 ; INPUTS:
                     00D6   471 ;       R2 = ADDRESS OF UNIT STATE VECTOR
                     00D6   472 ;       R3 = ADDRESS OF NEXT CHARACTER
                     00D6   473 ;       R4 = TWP ADDRESS
                     00D6   474 ;       R5 = UCB ADDRESS
                     00D6   475 ;
                     00D6   476 ; OUTPUTS:
                     00D6   477 ;
                     00D6   478 ;       R2 = ADDRESS OF UNIT STATE VECTOR
                     00D6   479 ;       R3 = ADDRESS OF NEXT CHARACTER
                     00D6   480 ;       R5 = UCB ADDRESS
                     00D6   481 ;       UCB$L_TT_OUTADR = ADDRESS OF START OF BURST
                     00D6   482 ;       UCB$L_TT_OUTLEN = LENGHT OF BURST
                     00D6   483 ;
                     00D6   484 ;                OR
                     00D6   485 ;       R3 = NEXT CHARACTER TO OUTPUT
                     00D6   486 ;--
                     00D6   487 FORMAT:
          50   DD    00D6   488          PUSHL   R0                              ; CSR MUST ALWAYS BE PRESERVED
                     00D8   489          IF_STATE <ESC_O>,FORMAT_X               ; IF ESCAPE IN PROGRESS, SPECIAL
 50   20 A4   53   C3 00DC   490          SUBL3   R3,TTY$L_WB_END(R4),R0          ; CALCULATE LENGTH OF STRING
   05 44 A5   09   E0 00F1   491          BBS     #TT$V_WRAP,UCB$L_DEVDEPEND(R5),3$; IF WE ARE A NO-WRAP
   14 44 A5   0C   E0 00E6   492          BBS     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),5$; SCOPE THEN OUTPUT THE MAXIMUM
                     00EB   493                                                  ; THAT WE CAN.
      42 A5   01   A3 00EB   494 3$:      SUBW3   #1,UCB$W_DEVBUFSIZ(R5),-        ; COMPUTE EOL-1
                   51        00EF   495          R1
   51   00FC C5   45   A2 00F0   496          SUBW2   UCB$W_TT_CURSOR(R5),R1          ; COMPUTE ROOM TILL EOL-1
                     1B   1B 00F5   497          BLEQU   FORMAT_X                        ; EOL REACHED
                     00F7   498
          51   51   32   00F7   499          CVTWL   R1,R1                           ;
          50   51   D1   00FA   500          CMPL    R1,R0                           ; OUTPUT SIZE IS LESS OF TWO
                     03   15 00FD   501          BLEQ    10$
          51   50   D0   00FF   502 5$:      MOVL    R0,R1                           ; ACTUAL IS LESS
                     0102   503
                     0102   504 10$:
          50   63   9A   0102   505          MOVZBL  (R3),R0                         ; TEST FIRST CHARACTER
            F0 8F   93   0105   506          BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
      0000'CF40          0108   507                  W^TTY$A_TYPE[R0]
                     2E   12 010C   508          BNEQ    FORMAT_X
                     010E   509
                     0E   BB 010E   510          PUSHR   #^M<R1,R2,R3>                   ; SAVE LENGTH, ADDRESS
0000'CF   63   51   2A   0110   511          SCANC   R1,(R3),W^TTY$A_TYPE,-          ; SKIP ALL NON SPECIALS
            F0 8F        0116   512          #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>
```

```
                          0118   513
                          0118   514 20$:
                    0E    BA    0118   515          POPR    #^M<R1,R2,R3>                   ; RESTORE LENGTH , ADDRESS
                51  50    C2    011A   516          SUBL    R0,R1                           ; COMPUTE NUMBER CHARS FOUND
                    1D    13    011D   517          BEQL    FORMAT_X                        ; YES
            00FC C5   51    A0    011F   518          ADDW    R1,UCB$W_TT_CURSOR(R5)          ; UPDATE FINAL CURSOR
                                0124   519                                                  ; THIS ASSUMES ALL CHARS
                                0124   520                                                  ; < SPACE ARE SPECIALS
                                0124   521          CLR_STATE <NL,SKIPLF,WRAP>              ;
                52  51    D0    0129   522          MOVL    R1,R2                           ; LENGTH OF STRING TO OUTPUT
                    50 8ED0   012C   523          POPL    R0                              ; RESTORE
            51  52    01    C3    012F   524          SUBL3   #1,R2,R1                        ; CALC N-1
        00FF C5   6341    90    0133   525          MOVB    (R3)[R1],UCB$B_TT_LASTC(R5)     ; LASTC IS FINAL CHARACTER OUTPUT
                  FF62    31    0139   526          BRW     STR_EXIT                        ; TAKE COMMON STRING EXIT
                                013C   527
                                013C   528 FORMAT_X:
                    50 8ED0   013C   529          POPL    R0                              ; RESTORE
                53  63    9A    013F   530          MOVZBL  (R3),R3                         ; GET CHARACTER
                  1C A4   D6    0142   531          INCL    TTY$L_WB_NEXT(R4)               ; BUMP NEXT POINTER
                    003F  31    0145   532          BRW     FORMAT_CHAR                     ; HANDLE AS SINGLE
```

```
                     0148   534          .SBTTL  OUTPUTANDWAIT - OUTPUT CHARACTER AND WAIT FOR INTERRUPT
                     0148   535 ;++
                     0148   536 ; OUTPUTANDWAIT - OUTPUT A CHARACTER AND WAIT INTERRUPT
                     0148   537 ;
                     0148   538 ; FUNCTIONAL DESCRIPTION:
                     0148   539 ;
                     0148   540 ; THIS ROUTINE IS USED BY THE OUTPUT INTERRUPT ROUTINES TO RETURN
                     0148   541 ; TO THE DEVICE DEPENDENT CODE TO OUTPUT A CHARACTER. THEIR RETURN
                     0148   542 ; CAUSES THE UNIT TO ENTER A WAIT FOR INTERRUPT STATE.
                     0148   543 ;
                     0148   544 ; INPUTS:
                     0148   545 ;
                     0148   546 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0148   547 ;       R3 = CHARACTER TO OUTPUT
                     0148   548 ;       R5 = UCB ADDRESS
                     0148   549 ;
                     0148   550 ; OUTPUTS:
                     0148   551 ;
                     0148   552 ;       THE UNIT IS PLACED IN A WAIT FOR INTERRUPT STATE
                     0148   553 ;       AND THE CONDITION CODES ARE SET TO PLUS INDICATING
                     0148   554 ;       SINGLE CHARACTER IN R3.
                     0148   555 ;
                     0148   556 ;       R5 = UCB ADDRESS
                     0148   557 ;--
                     0148   558 OUTPUTANDWAIT:                                ; OUTPUT CHARACTER AND WAIT ENTRY
00FF C5   53   90    0148   559          MOVB    R3,UCB$B_TT_LASTC(R5)       ; SAVE LAST CHARACTER OUTPUT
                     014D   560 OUTPUTANDWAIT1:
                     014D   561          TIMSET  1,,LOCKOUTPUT               ; ENABLE THE TIMER WITH A MINIMUM
                     0167   562                                              ; OF 1 SECOND AND INTERLOCK THE
                     0167   563                                              ; OUTPUT STREAM
010B C5   01   90    0167   564          MOVB    #1,UCB$B_TT_OUTYPE(R5)      ; SET THE KIND OF OUTPUT
               05    016C   565          RSB                                 ; AND RETURN WITH CHARACTER
                     016D   566
```

```
                     016D   568          .SBTTL  FORMAT_CHAR - FORMAT CHARACTER FOR OUTPUT
                     016D   569
                     016D   570          .ENABL  LSB
                     016D   571 ;++
                     016D   572 ; FORMAT_CHAR - FIND PROPER OUTPUT FOR SPECIFIED CHARACTER
                     016D   573 ;
                     016D   574 ; FUNCTIONAL DESCRIPTION:
                     016D   575 ;
                     016D   576 ; THIS ROUTINE TRANSLATES THE SPECIFIED CHARACTER FOR OUTPUT
                     016D   577 ; ON THE TARGET UNIT. THE OUTPUT OF THE SEQUENCE IS EITHER
                     016D   578 ; THE ORIGIONAL CHARACTER OR A STARTUP OF THE PROPER MULTIECHO
                     016D   579 ; STRING. CURSOR ADJUSTMENT IS ALSO DONE HERE FOR PRINTING
                     016D   580 ; CHARACTERS AND FORM CHARACTERS. IT IS POSSIBLE FOR THE RESULT TO BE NO OUTPUT.
                     016D   581 ;
                     016D   582 ; SEE EACH SPECIAL CHARACTER ROUTINE FOR MORE DETAILS ON ECHOING.
                     016D   583 ;
                     016D   584 ; INPUTS:
                     016D   585 ;
                     016D   586 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     016D   587 ;       R3 = CHARACTER TO TRANSLATE
                     016D   588 ;       R5 = UCB ADDRESS
                     016D   589 ;
                     016D   590 ; OUTPUTS:
                     016D   591 ;
                     016D   592 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     016D   593 ;       R3 = CHARACTER TO OUTPUT NEXT IF ANY
                     016D   594 ;       R5 = UCB ADDRESS
                     016D   595 ;--
                     016D   596 5$:     IF_NOT_STATE <ESC_O>,OUTPUTANDWAIT; BR IF NOT OUTPUT ESC
           034F   30 0171   597          BSBW    ESCSYNTAX_O             ; CHECK CHAR AGAINST ESC SYNTAX
             D2   14 0174   598          BGTR    OUTPUTANDWAIT           ; SYNTAX OK, OUTPUT CHARACTER
                     0176   599          IF_NOT_STATE <MULTI>,8$         ; IF MULTIECHOING AND EDITING
                     017A   600          IF_STATE <EDITING>,10$          ; DON'T ZERO CURSOR
        00FC C5   B4 017E   601 8$:     CLRW    UCB$W_TT_CURSOR(R5)     ; ZERO CURSOR (COLUMN) POSITION
     04 A2   0E   E4 0182   602 10$:    BBSC    #TTY$V_ST_ESC_O,4(R2),- ; SEQUENCE ENDED CORRECTLY
                  7A 0186   603                  GOOUT                   ; CLEAR STATE AND BRANCH ALWAYS
                     0187   604 FORMAT_CHAR:
                     0187   605          IF_STATE <PASALL,ESC_O,WRTALL>,5$       ; BR IF FORMAT NOT NEEDED
                     018F   606          CASE    W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
                     018F   607                          BSPACE,-
                     018F   608                          TAB,-
                     018F   609                          LINEFEED,-
                     018F   610                          VTAB,-
                     018F   611                          FORM,-
                     018F   612                          CARRIAGE,-
                     018F   613                          CTRLZ>
                     01A4   614
           A1 AF   9F 01A4   615          PUSHAB  OUTPUTANDWAIT           ; SET RETURN PC FOR ADJUST_CURSOR
                     01A7   616                                          ; AND FLOW INTO ADJUST_CURSOR
                     01A7   617
                     01A7   618          .DSABL  LSB
                     01A7   619
```

```
                                01A7   621          .SBTTL  ADJUST_CURSOR - Increment cursor and set wrap if needed
                                01A7   622
                                01A7   623 ADJUST_CURSOR:                              ; ADJUST CURSOR
                                01A7   624
                     FO 8F   93 01A7   625          BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
                0000'CF43       01AA   626          W^TTY$A_TYPE[R3]                   ; TEST FOR SPECIAL
                       28   12  01AE   627          BNEQ    30$                       ; NON SPACING CHARACTER
                   00FC C5   B6  01B0   628          INCW    UCB$W_TT_CURSOR(R5)       ; ADJUST CURSOR
              42 A5 00FC C5   B1  01B4   629          CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5) ; OVERRUN?
                       06   1E  01BA   630          BGEQU   5$                        ; YES
                                01BC   631
                                01BC   632 25$:     CLR_STATE -                       ; Set not at newline,
                                01BC   633                  <NL,SKIPLF,WRAP>          ; clear SKIPLF and WRAP.
                       05       01C1   634 26$:     RSB
                                01C2   635
                                01C2   636 5$:
                OC 44 A5   09   E0  01C2   637          BBS     #TT$V_WRAP,UCB$L_DEVDEPEND(R5),10$; BR IF WRAP ENABLED
                         F3   13  01C7   638          BEQL    25$                       ; IF EQL THEN AT END OF LINE AND OK
                EE 44 A5   0C   E0  01C9   639          BBS     #TT$V_SCOPE, -            ; IF SCREEN TERMINAL,CONTINUE
                                01CE   640                  UCB$L_DEVDEPEND(R5),25$ ;
                         8E   D5  01CE   641          TSTL    (SP)+                     ; DO NOT RETURN TO BSB TO DROP
                       00C6   31  01D0   642          BRW     DROP                      ; IF HARDCOPY,DROP CHARACTER
                                01D3   643 10$:     SET_STATE       CURSOR            ; SET CURSOR OVERFLOWED
                         E4   11  01D6   644          BRB     25$
                                01D8   645
        00000000'EF43   04   91  01D8   646 30$:     CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; IS THIS AN ESCAPE OR CSI
                         DF   12  01E0   647          BNEQ    26$                       ; NO THEN DON'T COUNT IT
                         8E   D5  01E2   648          TSTL    (SP)+                     ; DO NOT RETURN TO BSB TO DROP
                       00DE   31  01E4   649          BRW     ESCAPE                    ; *** FALL INTO THE ESCAPE CODE
```

```
                     01E7   651          .SBTTL  FORMAT_LOCAL - FORMAT CHARACTER FOR OUTPUT
                     01E7   652
                     01E7   653 ;++
                     01E7   654 ; FORMAT_LOCAL - FIND PROPER OUTPUT FOR SPECIFIED CHARACTER
                     01E7   655 ;
                     01E7   656 ; FUNCTIONAL DESCRIPTION:
                     01E7   657 ;
                     01E7   658 ; THIS ROUTINE TRANSLATES THE SPECIFIED CHARACTER FOR OUTPUT
                     01E7   659 ; ON THE TARGET UNIT FOR A NOECHO TERMINAL. THE OUTPUT OF THE
                     01E7   660 ; SEQUENCE MAY BE THE STARTUP OF THE PROPER MULTIECHO
                     01E7   661 ; STRING. OTHERWISE, CURSOR ADJUSTMENT IS ALSO DONE HERE FOR PRINTING
                     01E7   662 ; CHARACTERS AND FORM CHARACTERS.
                     01E7   663 ; IT IS POSSIBLE FOR THE RESULT TO BE NO OUTPUT.
                     01E7   664 ;
                     01E7   665 ; SEE EACH SPECIAL CHARACTER ROUTINE FOR MORE DETAILS ON ECHOING.
                     01E7   666 ;
                     01E7   667 ; INPUTS:
                     01E7   668 ;
                     01E7   669 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     01E7   670 ;       R3 = CHARACTER TO TRANSLATE
                     01E7   671 ;       R5 = UCB ADDRESS
                     01E7   672 ;
                     01E7   673 ; OUTPUTS:
                     01E7   674 ;
                     01E7   675 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     01E7   676 ;       R3 = CHARACTER TO OUTPUT NEXT IF ANY
                     01E7   677 ;       R5 = UCB ADDRESS
                     01E7   678 ;--
                     01E7   679
                     01E7   680 FORMAT_LOCAL:
                     01E7   681          CASE    W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
                     01E7   682                          BSPACE,-
                     01E7   683                          TAB_LOCAL,-
                     01E7   684                          LINEFEED,-
                     01E7   685                          VTAB,-
                     01E7   686                          FORM,-
                     01E7   687                          CARRIAGE,-
                     01E7   688                          CTRLZ>
                     01FC   689 ;
                     01FC   690 ; SPECIAL FORMATTING WAS NOT NEEDED - SIMPLY INCREMENT CURSOR
                     01FC   691 ; FOR THE LOCALLY-ECHOED CHARACTER AND CONTINUE.
                     01FC   692 ;
       A9    10      01FC   693          BSBB    ADJUST_CURSOR           ; ADJUST CURSOR
     FE04    31      01FE   694          BRW     GETNEXTCHAR             ; GET ANOTHER CHARACTER
     FF44    31      0201   695 GOOUT:   BRW     OUTPUTANDWAIT
```

```
                     0204   697          .SBTTL  CHARACTER OUTPUT FORMAT ACTION ROUTINES
                     0204   698          .SBTTL  BACKSPACE - output a backspace
                     0204   699
                     0204   700 ;++
                     0204   701 ; BACKSPACE - OUTPUT A BACKSPACE
                     0204   702 ;
                     0204   703 ; FUNCTIONAL DESCRIPTION:
                     0204   704 ;
                     0204   705 ; THIS ROUTINE OUTPUTS A BACKSPACE ON  A TERMINAL.
                     0204   706 ;
                     0204   707 ; INPUTS:
                     0204   708 ;
                     0204   709 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0204   710 ;       R3 = TTY$C_BS
                     0204   711 ;       R5 = UCB ADDRESS
                     0204   712 ;
                     0204   713 ; OUTPUTS:
                     0204   714 ;
                     0204   715 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0204   716 ;       R3 = TTY$C_BS
                     0204   717 ;       R4 = ADDRESS OF THE BS/SPACE/BS STRING IF APPROP.
                     0204   718 ;       R5 = UCB ADDRESS
                     0204   719 ;--
                     0204   720 BSPACE:                                      ; BACKSPACE
00FC C5    B7        0204   721          DECW    UCB$W_TT_CURSOR(R5)        ; ADJUST HORIZON
   FF3D    31        0208   722          BRW     OUTPUTANDWAIT              ; OUTPUT THE CHARACTER
```

```
                     020B   724          .SBTTL  CARRIAGE - format a carriage return
                     020B   725
                     020B   726 ;++
                     020B   727 ; CARRIAGE - FORMAT FOR CARRIAGE RETURN
                     020B   728 ;
                     020B   729 ; FUNCTIONAL DESCRIPTION:
                     020B   730 ;
                     020B   731 ; THIS ROUTINE SETS UP THE PROPER FILL FOR A CARRIAGE RETURN ON
                     020B   732 ; THE TARGET UNIT.
                     020B   733 ;
                     020B   734 ; INPUTS:
                     020B   735 ;
                     020B   736 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     020B   737 ;       R3 = TTY$C_CR
                     020B   738 ;       R5 = UCB ADDRESS
                     020B   739 ;
                     020B   740 ; OUTPUTS:
                     020B   741 ;
                     020B   742 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     020B   743 ;       R3 = TTY$C_CR
                     020B   744 ;       R5 = UCB ADDRESS
                     020B   745 ;--
                     020B   746 CARRIAGE:                                    ;
                     020B   747          CLR_STATE <NL>                     ; SET NOT AT NEW LINE.
                     020F   748 CROUTPUT:
                     020F   749          IF_STATE -                         ; If in a write state, branch
                     020F   750                  WRITE,115$                 ; forward.
                     0213   751          IF_NOT_STATE -                     ; If not in a read state, also
                     0213   752                  READ,115$                  ; branch forward.
                     0217   753          IF_STATE <MULTI,EDITREAD>,130$     ; not a character echo or wrap
      51    78 A5    D0   021E   754          MOVL    UCB$L_SVAPTE(R5),R1        ; GET THE PACKET ADDRESS
      54    30 A1    3C   0222   755          MOVZWL  TTY$W_RB_LINOFF(R1),R4     ; GET THE OFFSET TO THIS CHARACTER
2C A1    2C B144    9E   0226   756          MOVAB   @TTY$L_RB_LIN(R1)[R4],TTY$L_RB_LIN(R1); AND MAKE THIS THE NEW
            30 A1    B4   022C   757          CLRW    TTY$W_RB_LINOFF(R1)        ; LINE BEGININNG
            32 A1    B4   022F   758          CLRW    TTY$W_RB_LINREST(R1)       ; JUST MAKE SURE WE ARE AT THE END
3C 0122 C5    07    E1   0232   759          BBC     #TTY$V_PC_NOCRLF,UCB$W_TT_PRTCTL(R5),110$;SO JUST ECHO THE CHARACTER
                     0238   760          SET_STATE SKIPCF                   ; NO FREE LINEFEED
               41    11   023C   761          BRB     115$                       ; ELSE DON'T DO THE LINEFEED
                     023E   762
                     023E   763 130$:    IF_NOT_STATE CTRLR,110$            ;
                     0242   764 ;
                     0242   765 ; THE FOLLOWING CODE WILL INSERT A LINEFEED AFTER ANY CARRIAGE RETURN
                     0242   766 ; THAT IS THE LAST CHARACTER IN THE READ PROMPT STRING OR IN THE READ BUFFER
                     0242   767 ; ITSELF.  ANY <CR> CHARACTERS IN THE PROMPT STRING PRIOR TO THE LAST CHARACTER
                     0242   768 ; ARE NOT TOUCHED.
                     0242   769 ;
      51    78 A5    D0   0242   770          MOVL    UCB$L_SVAPTE(R5),R1        ; GET THE READ PACKET ADDRESS
      54    34 A1    3C   0246   771          MOVZWL  TTY$W_RB_PRMLEN(R1),R4     ; GET THE PROMPT LENGTH
   54    4A A144    9E   024A   772          MOVAB   TTY$A_RB_PRM(R1)[R4],R4    ; AND THIS IS THE ADDRESS
   54    00D8 C5    D1   024F   773          CMPL    UCB$L_TT_MULTI(R5),R4      ; IS THIS CR PAST END OF PROMPT?
               22    1E   0254   774          BGEQU   112$                       ; YES THEN FORCE LINEFEED
                     0256   775 ; In the case where we are sending a clear to end of line sequence (signaled
                     0256   776 ; by SKIPCRLF being set) then
                     0256   777 ; we don't want the linefeed to show up if there is one after a leading
                     0256   778 ; <CR>.  The case where there is a leading linefeed is handled by
                     0256   779 ; setting SKIPLF in the EDITREAD routine.
                     0256   780 ;
```

```
                                 0256   781          IF_NOT_STATE SKIPCRLF,115$      ; IF NO LINEFEEDS WANTED THEN
                                 025A   782          CLR_STATE <SKIPCRLF>            ; CLEAR NO LINEFEED
      54   78 A5   0000004B 8F   C1    025E   783          ADDL3   #TTY$L_RB_DATA+1,UCB$L_SVAPTE(R5),R4; GET THE ADDRESS OF THE
                54   00D8 C5   D1    0267   784          CMPL    UCB$L_TT_MULTI(R5),R4   ; FIRST CHARACTER AND SEE IF THIS IS IT
                               11   12    026C   785          BNEQU   115$                    ; NO THEN HANDLE NORMALY
                                 026E   786          SET_STATE <SKIPLF>              ; OTHERWISE SKIP THE LINEFEED
                               0B   11    0272   787          BRB     115$                    ; AND HANDLE NORMALY.
                                 0274   788
                                 0274   789 110$:
                                 0274   790          IF_STATE <SKIPCRLF>,115$
                                 0278   791 112$:    SET_STATE <SENDLF,SKIPLF>       ; SET STATE TO FORCE LF AND SKIP NEXT REAL O
                                 027F   792 115$:
                     00FC C5     B4    027F   793          CLRW    UCB$W_TT_CURSOR(R5)     ; RESET HORIZON
                   0D  00FF C5   91    0283   794          CMPB    UCB$B_TT_LASTC(R5), -   ; OPTIMIZATION: WAS LAST CHAR
                                 0288   795                  #TTY$C_CR               ; A CARRIAGE RETURN?
                               0F   13    0288   796          BEQL    DROP                    ; YES, DON'T OUTPUT ANOTHER
            0102 C5   00F6 C5   90    028A   797          MOVB    UCB$B_TT_CRFILL(R5),UCB$B_TT_FILL(R5); SET UP FILL
                               03   13    0291   798          BEQL    120$                    ; IF EQL THEN OUTPUT
                                 0293   799          SET_STATE FILL                  ; OTHERWISE OUTPUT
                           FEAF   31    0296   800 120$:    BRW     OUTPUTANDWAIT
                           FD69   31    0299   801 DROP:    BRW     GETNEXTCHAR             ; CONTINUE IN NEXT LOGICAL STATE
```

```
                                                C 6
                     029C   803          .SBTTL  CTRLZ - output control-Z
                     029C   804
                     029C   805 ;++
                     029C   806 ; CTRLZ - OUTPUT A CONTROL Z BASED ON THE OPERATION
                     029C   807 ;
                     029C   808 ; FUNCTIONAL DESCRIPTION:
                     029C   809 ;
                     029C   810 ; IF THE CURRENT OPERATION IS A READ AND THE ^Z IS A TERMINATOR THE ECHO ^Z.
                     029C   811 ;
                     029C   812 ; INPUTS:
                     029C   813 ;
                     029C   814 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     029C   815 ;       R3 = CONTROL Z
                     029C   816 ;       R5 = UCB ADDRESS
                     029C   817 ;
                     029C   818 ; OUTPUTS:
                     029C   819 ;
                     029C   820 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     029C   821 ;       R3 = CONTROL Z
                     029C   822 ;       R4 = ADDRESS OF THE MULTIECHO STRING FOR CONTROL Z IF APPROP.
                     029C   823 ;       R5 = UCB ADDRESS
                     029C   824 ;--
                     029C   825 CTRLZ:                                       ; CONTROL Z
                     029C   826          IF_STATE WRITE,300$                ; IF THERE IS A WRITE ACTIVE THEN
                     02A0   827                                             ; ECHO NORMALY
                     02A0   828          IF_NOT_STATE READ,300$             ; OUTPUT 26(8) IF NOT READ
                     02A4   829          IF_NOT_STATE EOL,300$              ; ^Z IF TERMINATOR
 54   00000000'EF    D0   02A8   830          MOVL    TTY$A_EXITECHO,R4      ; ADDRESS MULTIECHO STRING
    08 48 A5    1D    E1   02AF   831          BBC     #TT2$V_DECCRT,UCB$L_DEVDEPND2(R5),299$
                     02B4   832          SET_STATE NINTMULTI                ; DON'T ALLOW THIS TO BE INTERUPTED
      54    04 A4    DE   02B8   833          MOVAL   4(R4),R4               ; OUTPUT THE DECCRT CONTROL-Z
      54    64    D0   02BC   834 299$:    MOVL    (R4),R4                ; GET THE ACTUAL STRING ADDRESS
         0D9E    31   02BF   835          BRW     STRTMULTI              ; START OUTPUT
         FE83    31   02C2   836 300$:    BRW     OUTPUTANDWAIT
```

```
                     02C5   838          .SBTTL  ESCAPE - format an escape character
                     02C5   839
                     02C5   840 ;++
                     02C5   841 ; ESCAPE - FORMAT A ESCAPE BASED ON OPERATION AND TARGET UNIT
                     02C5   842 ;
                     02C5   843 ; FUNCTIONAL DESCRIPTION:
                     02C5   844 ;
                     02C5   845 ; THIS ROUTINE FORMATS ESCAPES.
                     02C5   846 ;
                     02C5   847 ; READ OPERATION:
                     02C5   848 ;
                     02C5   849 ;       A "$" IS ECHOED IF THE ESCAPE IS A TERMINATOR.
                     02C5   850 ;
                     02C5   851 ; WRITE OPERATION:
                     02C5   852 ;
                     02C5   853 ;       THE ESCAPE IS OUTPUT. ON TERMINALS WITH THE CHARACTERISTIC TT$M_ESCAPE,
                     02C5   854 ;       THE REMAINDER OF THE SEQUENCE IS CHECKED FOR SYNTACTIC CORRECTNESS.
                     02C5   855 ;
                     02C5   856 ; INPUTS:
                     02C5   857 ;
                     02C5   858 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     02C5   859 ;       R3 = ESCAPE
                     02C5   860 ;       R5 = UCB ADDRESS
                     02C5   861 ;
                     02C5   862 ; OUTPUTS:
                     02C5   863 ;
                     02C5   864 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     02C5   865 ;       R3 = ESCAPE OR "$"
                     02C5   866 ;       R5 = UCB ADDRESS
                     02C5   867 ;--
                     02C5   868 ESCAPE:                                 ; OUTPUT ESCAPE STRING
                     02C5   869          IF_NOT_STATE EOL,250$          ; ALSO IF NOT TERMINATOR
                     02C9   870          IF_STATE <MULTI,WRITE,EDITREAD>,250$; MULTIECHO THEN ASSUME SEQUENCE
          53   24    9A   02D0   871          MOVZBL  #TTY$C_DOLLAR,R3       ; OTHERWISE ECHO DOLLAR SIGN
             FED1    30   02D3   872          BSBW    ADJUST_CURSOR          ; ADJUST CURSOR
             FE6F    31   02D6   873          BRW     OUTPUTANDWAIT          ; AND OUTPUT
                     02D9   874 ;
                     02D9   875 ; OUTPUT ESCAPE ON WRITE OR CONTROL R
                     02D9   876 ;
                     02D9   877 250$:
    16 48 A5   18    E0   02D9   878          BBS     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),255$ ; BR IF ANSI TERMINAL
                     02DE   879          IF_STATE <ESCAPE>,255$         ; IS THIS ESCAPE MODE
    41 A5   40 8F    91   02E2   880          CMPB    #TT$_VT5X,UCB$B_DEVTYPE(R5) ;VT5X OR VT100 TERMINAL?
               02    1B   02E7   881          BLEQU   252$                   ; If LEQU, then maybe.
               23    11   02E9   882          BRB     260$                   ; Otherwise, no.
                     02EB   883
                     02EB   884 252$:
    41 A5   80 8F    91   02EB   885          CMPB    #TT$_VT100+32,UCB$B_DEVTYPE(R5);
               02    1A   02F0   886          BGTRU   255$                   ; If GTRU, then yes.
               1A    11   02F2   887          BRB     260$                   ; Otherwise, no.
                     02F4   888
                     02F4   889 255$:
                     02F4   890          SET_STATE <ESC_O>              ; OUTPUTTING AN ESCAPE SEQUENCE
               51    DD   02F9   891          PUSHL   R1                     ; SAVE R1
             019E    30   02FB   892          BSBW    ESCINIT                ; INIT THE ESCAPE SEQUENCE RULES
  0104 C5    51    90   02FE   893          MOVB    R1,UCB$B_TT_ESC_O(R5)  ; ...
               51 8ED0   0303   894          POPL    R1                     ; RESTORE R1
```

```
                     0306   895
                     0306   896          IF_NOT_STATE <MULTI>,260$       ; MULTIECHOING AND EDITING
                     030A   897          IF_STATE <EDITING>,270$         ; DON'T ZERO CURSOR
                     030E   898 260$:
00FC C5   B4         030E   899          CLR.    UCB$W_TT_CURSOR(R5)     ; ZERO CURSOR (COLUMN) POSITION
   FE33   31         0312   900 270$:    BRW     OUTPUTANDWAIT           ; FALL INTO OUTPUT AND WAIT
```

```
                     0315   902          .SBTTL  LINEFEED - format a line feed
                     0315   903
                     0315   904 ;++
                     0315   905 ; LINEFEED - FORMAT LINE FEED FOR TARGET UNIT
                     0315   906 ;
                     0315   907 ; FUNCTIONAL DESCRIPTION:
                     0315   908 ;
                     0315   909 ; THIS ROUTINE SETS UP THE PROPER FILL FOR A LINE FEED ON THE TARGET
                     0315   910 ; UNIT AND ADJUSTS THE CURSOR AND VERTICAL LINE COUNT.
                     0315   911 ;
                     0315   912 ; INPUTS:
                     0315   913 ;
                     0315   914 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0315   915 ;       R3 = TTY$C_LF
                     0315   916 ;       R5 = UCB ADDRESS
                     0315   917 ;
                     0315   918 ; OUTPUTS:
                     0315   919 ;
                     0315   920 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0315   921 ;       R3 = TTY$C_LF
                     0315   922 ;       R5 = UCB ADDRESS
                     0315   923 ;--
                     0315   924          .ENABLE LSB
                     0315   925 LINEFEED:
3B 04 A2    0D   E4  0315   926          BBSC    #TTY$V_ST_SKIPLF,4(R2),227$; SKIP AND SET OFF SKIP CONDITION
                     031A   927 ;
                     031A   928 ; NOTE THAT BECAUSE OF ACBB'S SIGNED BRANCH IT IS NOT APPROPRIATE HERE.
                     031A   929 ;
                     031A   930 LFOUTPUT:
        00FC C5      B5  031A   931          TSTW    UCB$W_TT_CURSOR(R5)     ; LINE FEED AT CURSOR 0 IS NEWLINE
             04      12  031E   932          BNEQ    210$                    ; IF NEQ THEN NO NL POSSIBLE
                     0320   933          SET_STATE NL                    ; SET NEW LINE
        00FE C5      96  0324   934 210$:    INCB    UCB$B_TT_LINE(R5)       ; ADJUST VERTICAL COUNT
47 A5   00FE C5      91  0328   935          CMPB    UCB$B_TT_LINE(R5),UCB$L_DEVDEPEND+3(R5);
             04      1F  032E   936          BLSSU   215$                    ; IF LSSU THEN NO PAGE CROSS
        00FE C5      94  0330   937          CLRB    UCB$B_TT_LINE(R5)       ; RESET VERTICAL POSITION
                     0334   938 215$:    IF_NOT_STATE WRITE,220$         ; BR IF NOT WRITE
   54   00D4 C5      D0  0338   939          MOVL    UCB$L_TT_WRTBUF(R5),R4  ; Address current write buffer.
     54   24 A4      D0  033D   940          MOVL    TTY$L_WB_IRP(R4),R4     ; Get associated IRP address.
             03      13  0341   941          BEQL    220$                    ; Branch if no IRP.
          38 A4      B6  0343   942          INCW    IRP$L_MEDIA(R4)         ; COUNT LINE
0102 C5 00F7 C5      90  0346   943 220$:    MOVB    UCB$B_TT_LFFILL(R5),UCB$B_TT_FILL(R5); INSERT TO CAUSE FILL
             03      13  034D   944          BEQL    225$                    ; OUTPUT IF NO FILL NEEDED
                     034F   945          SET_STATE FILL                  ; OUTPUT FILL CHARACTERS
            FDF3     31  0352   946 225$:    BRW     OUTPUTANDWAIT
            FF41     31  0355   947 227$:    BRW     DROP
                     0358   948          .DISABLE LSB
```

```
                                0358   950          .SBTTL  TAB - Output a tab.
                                0358   951
                                0358   952 ;++
                                0358   953 ; TAB - OUTPUT A TAB ON THE TARGET TERMINAL BASED ON CURSOR POSITION
                                0358   954 ;
                                0358   955 ; FUNCTIONAL DESCRIPTION:
                                0358   956 ;
                                0358   957 ; THIS ROUTINE IS ENTERED TO OUTPUT A TAB ON THE TARGET UNIT.
                                0358   958 ; IF THE TERMINAL HAS MECHANICAL TAB THEN THE TAB GOES DIRECT.
                                0358   959 ; OTHERWISE, THE CURSOR POSITION IS USED TO CALC. HOW MANY BLANKS
                                0358   960 ; TO OUTPUT TO MOVE THE CURSOR TO THE NEXT TAB STOP.
                                0358   961 ;
                                0358   962 ; INPUTS:
                                0358   963 ;
                                0358   964 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                                0358   965 ;       R3 = TTY$C_TAB
                                0358   966 ;       R5 = UCB ADDRESS
                                0358   967 ;
                                0358   968 ; OUTPUTS:
                                0358   969 ;
                                0358   970 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                                0358   971 ;       R3 = TTY$C_TAB
                                0358   972 ;       R4 = ADDRESS OF TAB MULTIECHO STRING IF APPROP.
                                0358   973 ;       R5 = UCB ADDRESS
                                0358   974 ;--
54   00FC C5   03   00   EF     0358   975 TAB:    EXTZV   #0,#3,UCB$W_TT_CURSOR(R5),R4; GET HORIZON POINTER
                                035F   976         IF_STATE TABEXPAND,525$            ; EXPAND THE TABS
        0C 44 A5   08   E0     0363   977         BBS     #TT$V_MECHTAB,UCB$L_DEVDEPEND(R5),530$; OUTPUT IF MECHANICAL HELP
         51   08   54   A3     0368   978 525$:   SUBW3   R4,#8,R1                    ;SETUP THE LENGTH OF THE TAB
         54   0001'CF   9E     036C   979         MOVAB   W^TTY$A_TAB+1,R4           ; ADDRESS STRING TO OUTPUT
                 0CFA   31     0371   980         BRW     STRTMULTI_1                ; START MULTIPLE OUTPUT
                                0374   981 ;
                                0374   982 ; MECHANICAL TAB TERMINAL
                                0374   983 ;
                                0374   984 ; THIS ROUTINE ASSUMES THAT THE TABS ARE SET ON 8'S
                                0374   985 ;
        41 A5   40 8F   91     0374   986 530$:   CMPB    #TT$_VT5X,UCB$B_DEVTYPE(R5); IN RANGE FOR SPECIAL CASE VT5X
                   18   1A     0379   987         BGTRU   535$                       ; IF GTR THEN NO
        41 A5   80 8F   91     037B   988         CMPB    #TT$_VT100+32,UCB$B_DEVTYPE(R5);
                   11   1B     0380   989         BLEQU   535$                       ;
53   42 A5   00FC C5   A3     0382   990         SUBW3   UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3; DISTANCE FROM RIGHT
                   05   1B     0389   991         BLEQU   534$                       ; IF LEQU THEN SKIP TEST
                53   08   B1     038B   992         CMPW    #8,R3                      ; IN LAST TAB SPACE?
                   D8   1E     038E   993         BGEQU   525$                       ; IF TRUE THEN SKIP MECH TAB
                53   09   9A     0390   994 534$:   MOVZBL  #TTY$C_TAB,R3             ;
          00FC C5   54   AA     0393   995 535$:   BICW    R4,UCB$W_TT_CURSOR(R5)     ; SETUP CURSOR STOP ON 8
          00FC C5   08   A0     0393   996         ADDW    #8,UCB$W_TT_CURSOR(R5)     ;
                                039D   997         CLR_STATE -                        ; No longer at newline; haven't
                                039D   998                 <NL,WRAP>                  ; just wrapped.
                    FDA3   31   03A2   999         BRW     OUTPUTANDWAIT              ;
                                03A5  1000 ;
                                03A5  1001 ; FOR LOCAL TABS, TRACK POSITION, BUT DON'T ECHO
                                03A5  1002 ;
                                03A5  1003
                                03A5  1004 TAB_LOCAL:
54   00FC C5   03   00   EF     03A5  1005         EXTZV   #0,#3,UCB$W_TT_CURSOR(R5),R4; GET HORIZON POINTER
          00FC C5   54   AA     03AC  1006         BICW    R4,UCB$W_TT_CURSOR(R5)     ; SETUP CURSOR STOP ON 8
```

```
                                                H  6

                00FC C5    08    A0  03B1  1007          ADDW    #8,UCB$W_TT_CURSOR(R5)  ;
                                     03B6  1008          CLR_STATE -
                                     03B6  1009                  <NL,WRAP>
                           FC47    31  03BB  1010          BRW     GETNEXTCHAR             ; GET NEXT OUTPUT CHARACTER
```

```
                     03BE  1012          .SBTTL  VTAB, FORM - output a vertical tab, or form feed
                     03BE  1013
                     03BE  1014 ;++
                     03BE  1015 ; VTAB - OUTPUT A VERTICAL TAB
                     03BE  1016 ; FORM - OUTPUT A FORM FEED
                     03BE  1017 ;
                     03BE  1018 ; FUNCTIONAL DESCRIPTION:
                     03BE  1019 ;
                     03BE  1020 ; THIS ROUTINE SETS UP THE PROPER OUTPUT STRING FOR A VTAB OF FORM FEED
                     03BE  1021 ; ON THE TARGET UNIT.  VTAB TRANSLATES TO 4 LINE FEEDS. AND FORM FEED
                     03BE  1022 ; TO MULTIPLE LINE FEEDS BASED ON THE PRESENCE OF MECHANICAL HELP
                     03BE  1023 ; AND THE CURRENT VERTICAL LINE POSITION. TO AVOID THE NECESSITY
                     03BE  1024 ; FOR A LARGE NUMBER OF LF DATA BYTES THE FORM FEED CAUSES A MULTI
                     03BE  1025 ; ECHO STRING OF 4 LINE FEEDS FOLLOWED BY ANOTHER FORM FEED TO BE
                     03BE  1026 ; OUTPUT UNTIL A PAGE CROSS TAKES PLACE.
                     03BE  1027 ;
                     03BE  1028 ; INPUTS:
                     03BE  1029 ;
                     03BE  1030 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     03BE  1031 ;       R3 = C_VTAB OR C_FF
                     03BE  1032 ;       R5 = UCB ADDRESS
                     03BE  1033 ;
                     03BE  1034 ; OUTPUTS:
                     03BE  1035 ;
                     03BE  1036 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     03BE  1037 ;       R5 = UCB ADDRESS
                     03BE  1038 ;--
          54 0000'CF 9E  03BE  1039 VTAB:   MOVAB   W^TTY$A_VTAB,R4          ; SET UP OUTPUT
             0C9A    31  03C3  1040         BRW     STRTMULTI                ; START MULTIPLE OUTPUT
                     03C6  1041 ;
                     03C6  1042 ; FORM FEED
                     03C6  1043 ;
       07 44 A5  13  E1  03C6  1044 FORM:  BBC     #TT$V_MECHFORM, -        ; BRANCH IF TERMINAL DOES NOT
                     03CB  1045                     UCB$L_DEVDEPEND(R5), -   ; SUPPORT MECHNICAL FORM FEEDS
                     03CB  1046                     505$                     ;
                00FE C5  94  03CB  1047         CLRB    UCB$B_TT_LINE(R5)        ; RESET LINE POSITION
                   FD76  31  03CF  1048         BRW     OUTPUTANDWAIT            ;
                      54 D4  03D2  1049 505$:   CLRL    R4                       ; SEND MULTIPLE LINE FEEDS
          54    47 A5 00FE C5  83  03D4  1050         SUBB3   UCB$B_TT_LINE(R5),UCB$L_DEVDEPEND+3(R5),R4; GET NUMBER TO END OF PAG
                      04  54  91  03DB  1051         CMPB    R4,#4            ; OUTPUT THE LF'S IN GROUPS OF 8
                      0C  1A  03DE  1052         BGTRU   520$             ; IF GTRU THEN MORE THAN 4 TO OUTPUT
                51    54  01  A1  03E0  1053 510$:   ADDW3   #1,R4,R1         ; GET THE NUMBER OF LINEFEEDS
                54  0001'CF  9E  03E4  1054         MOVAB   W^TTY$A_FORM+1,R4; GET THE ADDRESS OF THE STRING
                     0C82  31  03E9  1055         BRW     STRTMULTI_1      ; START OUTPUT
                54  0000'CF  9E  03EC  1056 520$:   MOVAB   W^TTY$A_LONGFORM,R4; GET ADDRESS TO STRING
                     0C6C  31  03F1  1057         BRW     STRTMULTI        ;
```

```
                                J 6
03F4  1059          .SBTTL  DISPATCHER SERVICE ROUTINES
```

```
                     03F4  1061          .SBTTL  BACKSPACING - OUTPUT N BACKSPACES
                     03F4  1062 ;++
                     03F4  1063 ; BACKSPACING:
                     03F4  1064 ;
                     03F4  1065 ; DESCRIPTION:
                     03F4  1066 ;       SEND A GIVEN NUMBER OF BACKSPACES TO THE TERMINAL
                     03F4  1067 ;
                     03F4  1068 ; INPUTS:
                     03F4  1069 ;       R2 = UNIT STATE VECTOR ADDRESS
                     03F4  1070 ;       R5 = UCB ADDRESS
                     03F4  1071 ;       UCB$W_TT_MULTILEN = NUMBER OF BACKSPACES TO OUTPUT
                     03F4  1072 ;--
                     03F4  1073 BACKSPACING:
0100 C5    B7        03F4  1074          DECW    UCB$W_TT_BSPLEN(R5)     ; SUBTRACT ONE AND OUTPUT THE BACKSPACE
     06    19        03F8  1075          BLSS    10$                     ; NO MORE THEN FINISH UP
53   08    9A        03FA  1076          MOVZBL  #TTY$C_BS,R3            ; SETUP BACKSPACE
   FD87    31        03FD  1077          BRW     FORMAT_CHAR             ; AND FORMAT THE CHARACTER
                     0400  1078 10$:     CLR_STATE BACKSPACE             ; CLEAR THE STATE WHEN DONE
   FBFF    31        0403  1079          BRW     GETNEXTCHAR             ; AND GET THE NEXT CHARACTER
                     0406  1080
```

```
                     0406  1082          .SBTTL  CURSOROVRFLOW - insert newline to handle end of line
                     0406  1083 ;++
                     0406  1084 ; CURSOROVRFLOW - INSERT A NEWLINE IN THE OUTPUT STREAM
                     0406  1085 ;
                     0406  1086 ; FUNCTIONAL DESCRIPTION:
                     0406  1087 ;
                     0406  1088 ; THIS ROUTINE IS ENTERED TO INJECT A CR/LF IN THE OUTPUT
                     0406  1089 ; STREAM REGARDLESS OF THE CURRENT STATE.
                     0406  1090 ;
                     0406  1091 ; INPUTS:
                     0406  1092 ;
                     0406  1093 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0406  1094 ;       R5 = UCB ADDRESS
                     0406  1095 ;
                     0406  1096 ; OUTPUTS:
                     0406  1097 ;
                     0406  1098 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0406  1099 ;       R3 = TTY$C_CR
                     0406  1100 ;       R4 = READ PACKET ADDRESS
                     0406  1101 ;       R5 = UCB ADDRESS
                     0406  1102 ;--
                     0406  1103 CURSOROVRFLOW:                                ;
                     0406  1104          CLR_STATE CURSOR                    ; CLEAR THE CONDITION
                     0409  1105          SET_STATE WRAP                      ; Signal WRAPping.
                     040E  1106          IF_NOT_STATE READ,99$               ; IF WE ARE NOT READING THEN HANDLE NORMALY
                     0412  1107          IF_STATE RDVERIFY,98$               ; NO CURSOR ORERFLOW ON READ VERIFY'S
          54   78 A5   D0  0416  1108          MOVL    UCB$L_SVAPTE(R5),R4    ; OTHERWISE GET THE PACKED ADDRESS
                     041A  1109          IF_NOT_STATE CTRLR,97$              ; IF WE ARE CONTROL-RING HANDLE NORMALY
       2C A4   00D8 C5   D1  041E  1110          CMPL    UCB$L_TT_MULTI(R5),TTY$L_RB_LIN(R4); IF WE AREN'T ON THE LAST LINE
                 05   1B  0424  1111          BLEQU   99$                     ; THEN DON'T WORRIE ABOUT LINREST
             32 A4    B5  0426  1112 97$:      TSTW    TTY$W_RB_LINREST(R4)    ; ARE WE IN THE MIDDLE OF A LINE
                 0D   12  0429  1113          BNEQ    98$                     ; YES THEN LEAVE THE STATUS QUOE
          53     0D   9A  042B  1114 99$:      MOVZBL  #TTY$C_CR,R3            ; SET FIRST CHARACTER IS CARRIAGE RETURN
                          042E  1115          SET_STATE <SENDLF,SKIPLF>       ; FORCE A FREE LINEFEED
             FDD7    31  0435  1116          BRW     CROUTPUT                ; CONTINUE WITH CARRIAGE RETURN CODE
             FE5E    31  0438  1117 98$:      BRW     DROP
```

```
                        1119          .SBTTL  EOLSEEN - handle end of line condition
                        1120
                        1121 ;++
                        1122 ; EOLSEEN - END OF LINE SEEN
                        1123 ;
                        1124 ; FUNCTIONAL DESCRIPTION:
                        1125 ;
                        1126 ; THIS ROUTINE IS ENTERED AFTER AN END OF LINE CHARACTER HAS
                        1127 ; TERMINATED A READ RECORD AND THE ECHO OF THAT CHARACTER IS COMPLETE.
                        1128 ;
                        1129 ; THE ACTION IS TO COMPLETE THE READ OPERATION AND IF NECESSARY,
                        1130 ; SEND AN XOFF.
                        1131 ;
                        1132 ; INPUTS:
                        1133 ;
                        1134 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        1135 ;       R5 = UCB ADDRESS
                        1136 ;
                        1137 ; OUTPUTS:
                        1138 ;
                        1139 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        1140 ;       R5 = UCB ADDRESS
                        1141 ;--
                        1142 EOLSEEN:                                   ; END READ OPERATION
                        1143 ;
                        1144 ; IN ORDER FOR THE FINAL CHARACTERS OF A READ VERIFY READ TO ECHO
                        1145 ; PROPERLY EDIT READ MUST BE ALLOWED TO COMPLETE, THIS CAN BE DONE
                        1146 ; IN TWO WAYS, FIRST CHANGING THE STATE DISPATCHER TO ALLOW EDIT READ
                        1147 ; TO HAVE HIGHER PRIORITY THEN EOL.  THIS FIXES EOL IN THE STATE TABLE
                        1148 ; FOR NOW AND FOREVER.  BY PUTTING THIS CODE HERE IT ALLOWS EOL TO
                        1149 ; MOVE TO ANY PLACE IN THE STATE DISPATCHER.
                        1150 ;
                        1151          IF_NOT_STATE RDVERIFY,5$          ; LET EDIT READ COMPLETE UNDER
                        1152          IF_NOT_STATE EDITREAD,5$          ; READ VERIFY
      0905    31        1153          BRW     EDITREAD                  ;...
                        1154
7C A5   01    9B        1155 5$:      MOVZBW  #SS$_NORMAL,UCB$W_BOFF(R5); SET STATUS
      FBB3'   30        1156          BSBW    TTY$READONE               ; COMPLETE THE I/O
      FBB0    31        1157          BRW     TTY$GETNEXTCHAR           ; CONTINUE
```

```
                     0450  1159          .SBTTL  FILLING - continue outputting fill characters
                     0450  1160
                     0450  1161 ;++
                     0450  1162 ; FILLING - CONTINUE FILL OUTPUT
                     0450  1163 ;
                     0450  1164 ; FUNCTIONAL DESCRIPTION:
                     0450  1165 ;
                     0450  1166 ; THIS ROUTINE IS ENTERED WHILE THE TRM$V_FILL STATE IS ON.
                     0450  1167 ; THE FILL COUNT IS DECREMENTED AND IF NON 0 A FILL IS OUTPUT
                     0450  1168 ; IF THE COUNT GOES TO 0 THEN THE FILL STATE IS CLEARED.
                     0450  1169 ;
                     0450  1170 ; INPUTS:
                     0450  1171 ;
                     0450  1172 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0450  1173 ;       R5 = UCB ADDRESS
                     0450  1174 ;
                     0450  1175 ; OUTPUTS:
                     0450  1176 ;
                     0450  1177 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0450  1178 ;       R3 = 0
                     0450  1179 ;       R5 = UCB ADDRESS
                     0450  1180 ;--
                     0450  1181 FILLING:                                     ; OUTPUT CONTINUING FILLS
0102 C5   97         0450  1182          DECB    UCB$B_TT_FILL(R5)          ; ADJUST FILL COUNT
     03   12         0454  1183          BNEQ    25$                        ; IF NEQ THEN NOT DONE YET
                     0456  1184          CLR_STATE FILL                     ; SET CONDITION OFF
     53   D4         0459  1185 25$:     CLRL    R3                         ; SET FILL CHARACTER
   FCEF   31         045B  1186          BRW     OUTPUTANDWAIT1             ; OUTPUT THE CHARACTER
```

```
                                045E  1188          .SBTTL  MULTIECHOING - Continue outputting multiecho sequence
                                045E  1189
                                045E  1190 ;++
                                045E  1191 ; MULTIECHOING - CONTINUE MULTIECHO STRING OUTPUT
                                045E  1192 ;
                                045E  1193 ; FUNCTIONAL DESCRIPTION:
                                045E  1194 ;
                                045E  1195 ; THIS ROUTINE IS ENTERED WHEN TTY$V_ST_MULTI IS SET. THE NEXT
                                045E  1196 ; CHARACTER IN THE STRING ADDRESSED BY UCB$L_TT_MULTI IS OUTPUT.
                                045E  1197 ; IF THE NEXT CHARACTER IS ZERO THEN THE STRING OUTPUT IS COMPLETE
                                045E  1198 ; AND THE MULTIECHO CONDITION IS RESET FOLLOWED BY A RETURN TO
                                045E  1199 ; THE OUTPUT STATE ANALYSIS ROUTINE IN GETNEXTCHAR.
                                045E  1200 ; IF A LENGTH IS SPECIFIED IN UCB$W_TT_MULTILEN, THEN IT IS USED TO DETECT
                                045E  1201 ; THE END OF STRING.
                                045E  1202 ;
                                045E  1203 ; INPUTS:
                                045E  1204 ;
                                045E  1205 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                                045E  1206 ;       R5 = UCB ADDRESS
                                045E  1207 ;
                                045E  1208 ;       UCB$W_TT_MULTILEN = ALTERNATE LENGTH
                                045E  1209 ;
                                045E  1210 ; OUTPUTS:
                                045E  1211 ;
                                045E  1212 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                                045E  1213 ;       R3 = CHARACTER TO OUTPUT ( FALL THROUGH TO ECHOFORMAT )
                                045E  1214 ;       R5 = UCB ADDRESS
                                045E  1215 ;--
                                045E  1216 MULTIECHOING:                                ; CONTINUE STRING MULTI ECHO
                00DC C5   B7    045E  1217          DECW    UCB$W_TT_MULTILEN(R5)       ; DECREMENT THE COUNT
                      23  18    0462  1218          BGEQ    20$                         ; CONTINUE IF MORE LEFT
        00D8 C5 00E0 C5   D0    0464  1219          MOVL    UCB$L_TT_SMLT(R5),UCB$L_TT_MULTI(R5); RESTORE SAVED ADDRESS
        00DC C5 00DE C5   B0    046B  1220          MOVW    UCB$W_TT_SMLTLEN(R5),UCB$W_TT_MULTILEN(R5);AND LENGTH
                      0B  14    0472  1221          BGTR    10$
                              0474  1222          CLR_STATE <MULTI,NINTMULTI>         ; NO MORE THEN RESET THE STATE
                    FB86  31    047C  1223          BRW     GETNEXTCHAR                 ; AND GET THE NEXT CHARACTER
                00DE C5   B4    047F  1224 10$:     CLRW    UCB$W_TT_SMLTLEN(R5)        ; CLEAR THE SAVED LENGTH
                00DC C5   B7    0483  1225          DECW    UCB$W_TT_MULTILEN(R5)       ; SUBTRACT OUT THIS CHARACTER
             53 00D8 D5   9A    0487  1226 20$:     MOVZBL  @UCB$L_TT_MULTI(R5),R3      ; GET NEXT MULTI ECHO CHARACTER
                00D8 C5   D6    048C  1227          INCL    UCB$L_TT_MULTI(R5)          ; ADJUST POINTER
                    FCF4  31    0490  1228          BRW     FORMAT_CHAR                 ;
```

```
                     0493  1230          .SBTTL  SENDLINEFEED - output a line feed
                     0493  1231
                     0493  1232 ;++
                     0493  1233 ; SENDLINEFEED - INSERT LINE FEED IN OUTPUT STREAM
                     0493  1234 ;
                     0493  1235 ; FUNCTIONAL DESCRIPTION:
                     0493  1236 ;
                     0493  1237 ; THIS ROUTINE IS USED TO INSERT A LINE FEED IN THE OUTPUT STREAM.
                     0493  1238 ;
                     0493  1239 ; INPUTS:
                     0493  1240 ;
                     0493  1241 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0493  1242 ;       R5 = UCB ADDRESS
                     0493  1243 ;
                     0493  1244 ; OUTPUTS:
                     0493  1245 ;
                     0493  1246 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0493  1247 ;       R3 = TTY$C_LF
                     0493  1248 ;       R5 = UCB ADDRESS
                     0493  1249 ;--
                     0493  1250 SENDLINEFEED:                           ; SEND OUT A LINE FEED
                     0493  1251          CLR_STATE SENDLF               ; CLEAR CONDITION
          53   0A   9A  0496  1252          MOVZBL  #TTY$C_LF,R3           ; SET TO OUTPUT LINE FEED
             FE7E   31  0499  1253          BRW     LFOUTPUT               ; CONTINUE IN LINE FEED CODE
```

```
                        049C  1255          .SBTTL  ESCAPE SEQUENCE PARSING SERVICES
                        049C  1256 ;++
                        049C  1257 ;
                        049C  1258 ; ESCINIT - INIT ESCAPE SEQUENCE RULES
                        049C  1259 ;
                        049C  1260 ; INPUTS:
                        049C  1261 ;       R3 = CHARACTER THAT IS INTRODUCER
                        049C  1262 ;
                        049C  1263 ; OUTPUTS:
                        049C  1264 ;       R1 = RULE TO START AT (BYTE)
                        049C  1265 ;
                        049C  1266 ESCINIT::
              51    D4  049C  1267          CLRL    R1                      ; USE THE NORMAL RULES
          80 8F     53  91  049E  1268          CMPB    R3,#^X80                        ; IF NOT AN 8 BIT CHARACTER
                    08  1F  04A2  1269          BLSSU   10$                     ;
     51 FFFFFF80'EF 43  90  04A4  1270          MOVB    TTY$A_ESCINIT-^X80[R3],R1; ELSE MOVE THRU THE TABLE
                        05  04AC  1271 10$:     RSB
```

```
                     04AD  1273          .SBTTL  ESCSYNTAX -- CHECK ESCAPE SEQUENCE SYNTAX
                     04AD  1274
                     04AD  1275 ;++
                     04AD  1276 ; FUNCTIONAL DESCRIPTION:
                     04AD  1277 ;       This routine checks the syntax of escape sequences, one
                     04AD  1278 ;       character at a time.
                     04AD  1279 ;
                     04AD  1280 ; INPUTS:
                     04AD  1281 ;       R3 contains the next character to be checked for correct syntax
                     04AD  1282 ;       UCB$B_TT_ESC(R5) contains a offset to the next syntax
                     04AD  1283 ;                rule in the TTY$A_ESCAPE syntax table for input data.
                     04AD  1284 ;       UCB$B_TT_ESC_O(R5) contains the offset for output data.
                     04AD  1285 ;       Assume R4 is available
                     04AD  1286 ;
                     04AD  1287 ; OUTPUTS:
                     04AD  1288 ;       Condition codes are set:
                     04AD  1289 ;       CC = POSITIVE means the character is syntactically correct
                     04AD  1290 ;       CC = EQUAL means syntactic correctness and sequence is complete
                     04AD  1291 ;       CC = NEGATIVE means error in parsing sequence
                     04AD  1292 ;
                     04AD  1293 ;       If CC = POSITIVE the UCB$B_TT_ESC (ESC_O) offset is updated.
                     04AD  1294 ;--
                     04AD  1295 ;
                     04AD  1296 ; INPUT ESCAPE SEQUENCES
                     04AD  1297 ;
                     04AD  1298
                     04AD  1299 ESCSYNTAX:
           54    DD  04AD  1300          PUSHL   R4                      ; SAVE R4
54   0103 C5    9A  04AF  1301          MOVZBL  UCB$B_TT_ESC(R5),R4     ; GET CURRENT STATE
                1A  10  04B4  1302          BSBB    E_SYNTAX
0103 C5      54  90  04B6  1303          MOVB    R4,UCB$B_TT_ESC(R5)     ; SAVE STATE
             54 8ED0  04BB  1304          POPL    R4                      ; AND RESTORE R4
          0103 C5    95  04BE  1305          TSTB    UCB$B_TT_ESC(R5)        ; SET THE CONDITION CODES
                05  04C2  1306          RSB
                     04C3  1307
                     04C3  1308 ;
                     04C3  1309 ; OUPUT ESCAPE SEQUENCES
                     04C3  1310 ;
                     04C3  1311
                     04C3  1312 ESCSYNTAX_O:
54   0104 C5    9A  04C3  1313          MOVZBL  UCB$B_TT_ESC_O(R5),R4   ; GET CURRENT STATE
                06  10  04C8  1314          BSBB    E_SYNTAX
0104 C5      54  90  04CA  1315          MOVB    R4,UCB$B_TT_ESC_O(R5)   ; SAVE CURRENT STATE
                05  04CF  1316          RSB
                     04D0  1317
                     04D0  1318
                     04D0  1319 ;
                     04D0  1320 ; GENERAL SEQUENCE HANDLER
                     04D0  1321 ;
                     04D0  1322
                     04D0  1323 E_SYNTAX::
                     04D0  1324
0000'CF44    53  91  04D0  1325 10$:    CMPB    R3, W^TTY$A_ESCAPE[R4]  ; check range of character
                0F  1F  04D6  1326          BLSSU   20$                     ; branch if not this rule
0001'CF44    53  91  04D8  1327          CMPB    R3, W^TTY$A_ESCAPE+1[R4]; lower than high limit?
                07  1A  04DE  1328          BGTRU   20$                     ; look to next rule
                     04E0  1329
```

```
                54   0002'CF44   9A  04E0  1330          MOVZBL  W^TTY$A_ESCAPE+2[R4],-  ; character is valid
                                     04E6  1331                  R4                     ; save offset to next rule
                                     04E6  1332
                                 05  04E6  1333          RSB                            ; offset=POS => next rule exists
                                     04E7  1334                                         ; offset=0 => end of sequence
                                     04E7  1335
                                     04E7  1336 ;
                                     04E7  1337 ; Continue looking for correct sequence rule
                                     04E7  1338 ;
                0002'CF44    95  04E7  1339 20$:    TSTB    W^TTY$A_ESCAPE+2[R4]    ; any next rule?
                      05     13  04EC  1340         BEQL    30$                     ; no; bad syntax
                  54  03     C0  04EE  1341         ADDL    #3, R4                  ; offset to next rule
                      DD     11  04F1  1342         BRB     10$                     ; continue syntax check
                                 04F3  1343 30$:
                  54  01     CE  04F3  1344         MNEGL   #1, R4                  ; error in sequence
                                 05  04F6  1345         RSB
```

```
                     04F7  1347          .SBTTL MOVEREADATA - MOVE CHARACTER FROM TYPEAHEAD TO READ BUFFER
                     04F7  1348 ;++
                     04F7  1349 ; MOVEREADATA -- MOVE CHARACTER FROM TYPEAHEAD BUFFER TO READ BUFFER
                     04F7  1350 ;
                     04F7  1351 ; FUNCTIONAL DESCRIPTION:
                     04F7  1352 ;
                     04F7  1353 ; THIS ROUTINE MOVES A CHARACTER FROM THE TYPEAHEAD BUFFER AND STARTS
                     04F7  1354 ; THE ECHO.
                     04F7  1355 ;
                     04F7  1356 ; NON-IMMEDIATE ACTION CONTROL SEQUENCES ARE HANDLED HERE.
                     04F7  1357 ;
                     04F7  1358 ; BEFORE RETURNING A CHARACTER FOR ECHO IT IS CONVERTED TO ITS
                     04F7  1359 ; MULTIPLE ECHO STRING IF APPROPRIATE. IN THIS CASE THE CHARACTER
                     04F7  1360 ; RETURNED IS THE FIRST OF THE MULTIPLE ECHO CHARACTERS.
                     04F7  1361 ;
                     04F7  1362 ; INPUTS:
                     04F7  1363 ;       R5 = UCB ADDRESS
                     04F7  1364 ;
                     04F7  1365 ; OUTPUTS:
                     04F7  1366 ;
                     04F7  1367 ;       R3 = CHARACTER IF ANY (CC = EQL )
                     04F7  1368 ;       R5 = UCB ADDRESS
                     04F7  1369 ;--
                     04F7  1370 ;
                     04F7  1371 ; GOTO TABLE
                     04F7  1372 ;
      011F   31      04F7  1373 TAB_NOTA:       BRW     NOTA
51    58 A5  D0      04FA  1374 TAB_EOLSEEN:    MOVL    UCB$L_IRP(R5),R1
      38 A1  B4      04FE  1375                 CLRW    IRP$L_MEDIA(R1)
      FF37   31      0501  1376                 BRW     EOLSEEN
      0153   31      0504  1377 TAB_DISMISS:    BRW     DISMISS
      0101   31      0507  1378 TAB_BUFEMPTY:   BRW     BUFEMPTY
      022E   31      050A  1379 TAB_INDELST:    BRW     INDELST
      0C1B   31      050D  1380 TAB_PASSALL:    BRW     PASSALL
      FAF2   31      0510  1381 TAB_GETNEXT:    BRW     GETNEXTCHAR
      02D8   31      0513  1382 TAB_QUOTE:      BRW     QUOTE
      02EA   31      0516  1383 TAB_EDITINGCHAR:BRW     EDITINGCHAR
        53   95      0519  1384 TAB_ESCINPROG:  TSTB    R3                  ; NO NULLS IN ESCAPE SEQUENCES
        E7   13      051B  1385                 BEQL    TAB_DISMISS
      013F   31      051D  1386                 BRW     ESCINPROG
      012A   31      0520  1387 TAB_CVTLOW:     BRW     CVTLOW
      022C   31      0523  1388 TAB_OPEN:       BRW     OPEN
                     0526  1389 ;
                     0526  1390 ; MAIN LINE MOVE OUT OF BUFFER CODE
                     0526  1391 MOVEREADATA:                                ;
                     0526  1392          .ENABLE LSB
   54 78 A5  D0      0526  1393          MOVL    UCB$L_SVAPTE(R5),R4       ; SETUP R4 TO CONTAIN THE ADDRESS OF THE
      14 A4  D4      052A  1394          CLRL    TTY$L_RB_ECHSTR(R4)       ; CLEAN OUT ANY UNUSED ECHO STRINGS
40 A4 3C A4  B1      052D  1395          CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); TEST BUFFER FULL
        C6   1E      0532  1396          BGEQU   TAB_EOLSEEN               ; BUFFER FULL THEN END
                     0534  1397 ;
                     0534  1398 ; MOVE CHARACTER OUT OF TYPEAHEAD BUFFER
                     0534  1399 ;
   54 00E4 C5  D0      0534  1400          MOVL    UCB$L_TT_TYPAHD(R5),R4    ; ADDRESS TYPEAHEAD BUFFER
        BC   13      0539  1401          BEQL    TAB_NOTA                  ; NO BUFFER THEN THERE IS A FORK
                     053B  1402                                            ; PROCESS WAITING SO LET IT GO.
      0C A4  B7      053B  1403          DECW    TTY$W_TA_INAHD(R4)        ; take the character out of typeahead
```

```
                C7   19  053E  1404          BLSS    TAB_BUFEMPTY            ; IF EQL THEN NO
          53    04 B4   9A  0540  1405          MOVZBL  @TTY$L_TA_GET(R4),R3    ; GET THE CHARACTER
                        0544  1406          IF_STATE DEL,TAB_INDELST        ; ARE WE IN DELETE STATE
                        0548  1407 ;
                        0548  1408 ; ACTUALLY REMOVE CHARACTER FROM BUFFER
                        0548  1409 ;
                        0548  1410 ;
                        0548  1411 ; CHECK FOR BUFFER WRAP AROUND
                        0548  1412 ;
  06 04 A4    10 A4     F2  0548  1413 40$:    AOBLSS  TTY$L_TA_END(R4),TTY$L_TA_GET(R4),60$; POINTER PAST END?
    04 A4   0118 C4     9E  054E  1414          MOVAB   TTY$L_TA_DATA(R4),TTY$L_TA_GET(R4); RESET POINTER
                        0554  1415 ;
                        0554  1416 ; SKIP TESTS FOR SECOND CONTROL SEQUENCE SET IF PASSALL
                        0554  1417 ;
                        0554  1418
          54    78 A5     D0  0554  1419 60$:    MOVL    UCB$L_SVAPTE(R5),R4     ; SETUP R4 TO CONTAIN THE ADDRESS OF THE
                        0558  1420          IF_STATE <PASSALL,NOFLTR>,TAB_PASSALL; IN PASSALL THEN PASS ALL
                        055F  1421          IF_STATE QUOTING,TAB_QUOTE      ; ARE WE QUOTING
  51  00000000'EF43     9A  0563  1422 67$:    MOVZBL  TTY$A_CCLIST[R3],R1     ; GET THE TOKEN
                A9      12  056B  1423          BNEQ    TAB_EDITINGCHAR         ; SPECAIL ACTION THEN HANDLE SPECAILY
                        056D  1424 ;
                        056D  1425 ; INSERT THE CHARACTER
                        056D  1426 ;
                        056D  1427 151$:   IF_STATE <ESC>,TAB_ESCINPROG; IF IN AN ESCAPE SEQUENCE THEN HANDLE AS SUCH
                        0571  1428 INSERT_CHAR:
          51    58 A5     D0  0571  1429 152$:   MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
    A6 20 A1    08     E0  0575  1430          BBS     #IO$V_CVTLOW,IRP$W_FUNC(R1),TAB_CVTLOW; BR IF CONVERT LOWER TO UPPER
    3D 1C B4    53     E0  057A  1431 155$:   BBS     R3,@TTY$L_RB_TERM(R4),TERMFOUND; IS THIS CHARACTER A TERMINATOR
          51    30 A4     3C  057F  1432          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET INTO THE BUFFER
                32 A4     B5  0583  1433          TSTW    TTY$W_RB_LINREST(R4)    ; DO WE HAVE TO OPEN THE BUFFER
                   9B     12  0586  1434          BNEQ    TAB_OPEN                ; NO THEN HANDLE NORMALLY
     2C B441    53     90  0588  1435 159$:   MOVB    R3,@TTY$L_RB_LIN(R4)[R1]; INSERT CHARACTER
                3C A4     B6  058D  1436          INCW    TTY$W_RB_TXTOFF(R4)     ; ADJUST POINTER
                30 A4     B6  0590  1437          INCW    TTY$W_RB_LINOFF(R4)     ; ADD TO COUNT
                        0593  1438 ;
                        0593  1439 ; CHECK FOR FULL BUFFER
                        0593  1440 ;
                        0593  1441 CHECK_BUFFER:
  40 A4     3C A4     B1  0593  1442 210$:   CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); TEST BUFFER FULL
                13      1E  0598  1443          BGEQU   212$                    ; BUFFER FULL THEN END
                        059A  1444 ;
                        059A  1445 ; ECHO THE CHARACTER IF NECESSARY
                        059A  1446 ;
                        059A  1447 230$:   IF_STATE        <NOECHO,ESC,EDITREAD>,250$; IF NOECHO OR ESC THEN BRANCH
                FBDD    31  05A7  1448          BRW     FORMAT_CHAR             ;
                FA58    31  05AA  1449 240$:   BRW     GETNEXTCHAR             ; VECTOR TO TRANSFER MORE CHARACTERS
                        05AD  1450 ;
                        05AD  1451 ; BUFFER OVERFLOWED THEN TERMINATE
                        05AD  1452 ;
          51    58 A5     D0  05AD  1453 212$:   MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
                38 A1     B4  05B1  1454          CLRW    IRP$L_MEDIA(R1)         ; SET NO TERMINATOR
  20 A1     1000 8F     AA  05B4  1455          BICW    #IO$M_TRMNOECHO,IRP$W_FUNC(R1); FORCE POSSIBLE ECHO
                18      11  05BA  1456          BRB     221$                    ; AND CONTINUE
                        05BC  1457 ;
                        05BC  1458 ; TERMINATOR FOUND BUT BUFFER NOT FULL
                        05BC  1459 ;
                        05BC  1460 TERMFOUND:
```

```
                51    58 A5    D0  05BC  1461          MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
                38 A1    53    B0  05C0  1462          MOVW    R3,IRP$L_MEDIA(R1)      ; SAVE LAST CHARACTER FOR STATUS
                   3A A1      B6  05C4  1463          INCW    IRP$L_MEDIA+2(R1)       ; SET TERMINATOR SIZE
                51    3C A4    3C  05C7  1464          MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE OFFSET TO THE LAST CHARACTER
             00 B441    53    90  05CB  1465          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]; PUT THE LAST CHARACTER AWAY
                51    58 A5    D0  05D0  1466          MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS BACK
                                 05D4  1467 221$:       SET_STATE <EOL>                 ; SET END OF LINE SEEN
          CD 20 A1    0C       E0  05D8  1468 225$:    BBS     #IO$V_TRMNOECHO,IRP$W_FUNC(R1),240$; BR IF NOT TERM NOECHO
                   FFBA    31    05DD  1469          BRW     230$
                                 05E0  1470 ;
                                 05E0  1471 ; TEST FOR ECHO NEEDS
                                 05E0  1472 ;
                                 05E0  1473 250$:    IF_STATE <EDITREAD,ESC>,240$    ; CONTINUE IF ESCAPE
                      00    E1   05ED  1474          BBC     #TT2$V_LOCALECHO,-
                   B8 48 A5       05EF  1475                  UCB$L_DEVDEPND2(R5),240$ ; NO FORMATTING IF NOT LOCAL ECHO
                51    58 A5    D0  05F2  1476          MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
          AF 20 A1    06       E0  05F6  1477          BBS     #IO$V_NOECHO, -
                                 05FB  1478                  IRP$W_FUNC(R1),240$     ; OR READ NOECHO
                      FBE9    31  05FB  1479          BRW     FORMAT_LOCAL            ; OTHERWISE, LOCAL ECHO TERMINAL, FORMAT
```

```
                                05FE  1481          .SBTTL  MOVEREADATA SERVICE ROUTINES
                                05FE  1482 ;
                                05FE  1483 ; TYPEAHEAD BUFFER EXHAUSTED
                                05FE  1484 ;
                                05FE  1485 25$:     IF_NOT_STATE NOECHO,21$          ; NO ECHO READ NECESSARY TO OPTOMIZE
                                0602  1486          SET_STATE PRE                    ; SET THE BYPASS TYPEAHED BUFFER STATE
                         0D  11 0606  1487          BRB     21$                      ; CONTINUE IN THE NORMAL CODE PATH
                                0608  1488 ;
                       0027  31 0608  1489 20$:     BRW     XON                      ; CONTINUE
                                060B  1490 BUFEMPTY:
                     0C A4  B4 060B  1491          CLRW    TTY$W_TA_INAHD(R4)       ; FIX THE COUNT TO REFLECT 0
                                060E  1492          IF_STATE <PASALL,NOFLTR>,25$     ;
                                0615  1493 21$:     IF_STATE <TYPFUL>,20$            ; BR IF TYPEAHEAD FULL
                                0619  1494 NOTA:
                                0619  1495 ;
                                0619  1496 ; THE TYPEAHEAD BUFFER HAS NOW BEEN EMPTIED INTO THE SYSTEM BUFFER.
                                0619  1497 ; IF PROCESSING A READ WITH ZERO SECOND TIMEOUT, RETURN THE DATA
                                0619  1498 ; TO THE USER IMMEDIATELY.
                                0619  1499 ;
           3C 68 A5    01   E1 0619  1500          BBC     #UCB$V_TT_TIMO,UCB$W_DEVSTS(R5),DISMISS; BR IF NOT READ WITH TIMEOUT
      00B0 C5    00000000'EF   D1 061E  1501          CMPL    TTY$A_MAXTIME,UCB$L_TT_RDUE(R5); ZERO SECOND TIMEOUT?
                          31   12 0627  1502          BNEQ    DISMISS                  ; BR IF NOT.
         7C A5    022C 8F   B0 0629  1503          MOVW    #SS$_TIMEOUT,UCB$W_BOFF(R5); SET TIMEOUT COMPLETION STATUS.
                       F9CE'  30 062F  1504          BSBW    TTY$READONE              ; COMPLETE REQUEST.
                              0632  1505
```

```
                     0632  1507 ;++
                     0632  1508 ; XON - SEND XON
                     0632  1509 ;
                     0632  1510 ; FUNCTIONAL DESCRIPTION:
                     0632  1511 ;
                     0632  1512 ; THIS ROUTINE IS USED TO SEND AN XON
                     0632  1513 ;
                     0632  1514 ; INPUTS:
                     0632  1515 ;
                     0632  1516 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0632  1517 ;       R5 = UCB ADDRESS
                     0632  1518 ;
                     0632  1519 ; OUTPUTS:
                     0632  1520 ;
                     0632  1521 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0632  1522 ;       R3 = XON
                     0632  1523 ;       R5 = UCB ADDRESS
                     0632  1524 ;--
                     0632  1525 XON:                                    ; SEND AN XON
                     0632  1526         CLR_STATE <TYPFUL>
                     0636  1527         IF_NOT_STATE <OVRFLO>,10$       ; SKIP IF NO OVERFLOW ERROR
                     063A  1528
                     063A  1529         CLR_STATE <OVRFLO>              ; RESET CONDITION
7C A5  0838 8F   B0  063E  1530         MOVW    #SS$_DATAOVERUN,JCB$W_BOFF(R5); COMPLETE READ IN ERROR
          F9B9'  30  0644  1531         BSBW    TTY$READONE
                     0647  1532
          F9B6'  30  0647  1533 10$:    BSBW    TTY$XON
          F9B8   31  064A  1534         BRW     GETNEXTCHAR
```

```
                                064D  1536
                                064D  1537 ;
                                064D  1538 ; CONVERT CHARACTER TO UPPER CASE
                                064D  1539 ;
03 0000'CF43   03   E1          064D  1540 CVTLOW: BBC     #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],154$; BR IF NOT LOWER CHARACTER
              53   20   AA      0654  1541          BICW    #^X020,R3               ; CONVERT TO UPPER CASE
                 FF20   31      0657  1542 154$:    BRW     155$
```

```
                     065A  1544 ;
                     065A  1545 ; DISMISS INTERRUPT
                     065A  1546 ;
                     065A  1547 DISMISS:
010B C5   94         065A  1548          CLRB    UCB$B_TT_OUTYPE(R5)     ; SET NO RETURN CHARACTER.
          05         065E  1549          RSB                             ; and dismiss the interupt
```

```
                     065F  1551 ;
                     065F  1552 ; ESCAPE SEQUENCE IN PROGRESS - CHECK SYNTAX
                     065F  1553 ;
                     065F  1554 ESCINPROG:
          53   95    065F  1555          TSTB    R3                          ; NO NULLS IN ESCAPE SEQUENCES
          4C   13    0661  1556          BEQL    961$                        ; SO DROP IT
    51 3C A4   3C    0663  1557          MOVZWL  TTY$W_RB_TXTOFF(R4),R1      ; GET THE PLACE TO STORE THE ESCAPE
 00 B441   53   90    0667  1558          MOVB    R3,@TTY$[_RB_TXT(R4)[R1]    ; AND PLACE THE CHARACTER IN THE RIGHT PLACE
       3C A4   B6    066C  1559          INCW    TTY$W_RB_TXTOFF(R4)         ; AND INCREMENT THE WORLD
    51 58 A5   D0    066F  1560          MOVL    UCB$L_IRP(R5),R1            ; GET THE IRP ADDRESS
       3A A1   B6    0673  1561          INCW    IRP$L_MEDIA+2(R1)           ; ADJUST SIZE OF ESCAPE SEQUENCE
        FE34   30    0676  1562          BSBW    ESCSYNTAX                   ; CHECK CHAR AGAINST SYNTAX
          2D   14    0679  1563          BGTR    960$                        ; VALID CHAR, CONTINUE INPUT
          04   13    067B  1564          BEQL    197$                        ; VALID CHAR, SEQUENCE COMPLETE
                     067D  1565 ;
                     067D  1566 ; ESCAPE SYNTAX ERROR
                     067D  1567 ;
                     067D  1568 195$:    SET_STATE BADESC                    ; SET BAD ESCAPE
                     0681  1569 197$:    CLR_STATE ESC                       ; SET SEQUENCE DONE
3C A4   3A A1   A2   0686  1570          SUBW    IRP$L_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); ADJUST TRANSFER SIZE
                     068B  1571          IF_STATE <NOFLTR,BADESC>,199$       ; BAD SEQUENCE THEN HANDLE NORMALY OR
                     0693  1572                                              ; NOFILTER READ
    53   7E 8F   91   0693  1573          CMPB    #^A/~/,R3                   ; IS THIS A FUNCTION KEY
          4C   12    0697  1574          BNEQ    910$                        ; NO THEN HANDLE APROPRIATELY
    48 A4   15   B1   0699  1575          CMPW    #21,TTY$W_RB_ESCTKN(R4)     ; IS THIS THE EXIT KEY
          13   12    069D  1576          BNEQ    920$                        ; NO THEN TRY NORMAL KEY
       53   1A   9A   069F  1577          MOVZBL  #TTY$C_CTRLZ,R3             ; SETUP A CONTROL-Z
       38 A1   D4    06A2  1578          CLRL    IRP$L_MEDIA(R1)             ; CLEAN THE ESCAPE SEQUENCE UP
        018B   31    06A5  1579          BRW     162$                        ;AND GO OFF TO DISPATCH
                     06A8  1580
         0078   31   06A8  1581 960$:    BRW     198$
                     06AB  1582
                     06AB  1583 199$:
                     06AB  1584          SET_STATE EOL                       ; SET UP END OF LINE
          F953   31   06AF  1585 961$:    BRW     GETNEXTCHAR                 ; THEN GET THE NEXT CHARACTER
                     06B2  1586
                     06B2  1587 ;
                     06B2  1588 ; FUNCTION KEY DETECTED CHANGE FUNCTION KEY TO TOKEN
                     06B2  1589 ; BUT ONLY IF WE ARE EDITING
                     06B2  1590 ;
                     06B2  1591 920$:    IF_NOT_STATE <EDITING>,199$         ; NO EDITING THE GO AND RETURN THE SEQUENDE
 0000'8F   48 A4   B1   06B6  1592          CMPW    TTY$W_RB_ESCTKN(R4),#TTY$K_MAXESCTKN; IS IT OUT OF RANGE
          57   1A    06BC  1593          BGTRU   190$                        ; YES THEN RETURN IT
       51   48 A4   3C   06BE  1594          MOVZWL  TTY$W_RB_ESCTKN(R4),R1      ; GET THE OFFSET
51 00000000'EF41   9A   06C2  1595          MOVZBL  TTY$A_FCNTKN[R1],R1         ; AND MAKE THAT INTO A TOKEN
                     06CA  1596 930$:
          51   95    06CA  1597          TSTB    R1                          ; IS THIS VALID
          47   13    06CC  1598          BEQL    190$                        ; NO THEN RETURN THE SEQUENCE
    53   58 A5   D0   06CE  1599 191$:    MOVL    UCB$L_IRP(R5),R3            ; GET THE IRP ADDRESS
       38 A3   D4    06D2  1600          CLRL    IRP$L_MEDIA(R3)             ; CLEAR THE ESCAPE SEQUENCE OUT
       0D   51   91   06D5  1601          CMPB    R1,#TTY$K_ET_UNUSED         ; IS THIS AN UNUSED ESCAPE SEQUENCE
          3F   13    06D8  1602          BEQL    970$                        ; YES THEN IGNORE IT...
       0B   51   91   06DA  1603          CMPB    R1,#TTY$K_ET_RECALL         ; IS THIS A RECALL COMMAND
          03   12    06DD  1604          BNEQ    192$                        ; NO THEN CONTINUE NORMALY
       53   02   9A   06DF  1605          MOVZBL  #TTY$C_CTRLB,R3             ; YES THEN MAKE IT A CONTROL-B AND CONTINUE
         0127   31   06E2  1606 192$:    BRW     70$
                     06E5  1607
```

```
                     06E5  1608 910$:   IF_NOT_STATE <EDITING>,199$      ; AND WE ARE NOT EDITING DO THE SAME
03   3A A1   B1     06E9  1609          CMPW    IRP$L_MEDIA+2(R1),#3     ; IS THE SEQUENCE LONGER THAN 3 CHARACTERS
          26   14     06ED  1610          BGTR    190$                     ; YES THEN DO A MATCH ON IT
   51     05   9A     06EF  1611          MOVZBL  #TTY$K_ET_BACK_CHAR,R1   ; SETUP FOR BACK CHARCTER
53    44 8F   91     06F2  1612          CMPB    #^A/D/,R3                ; IS IT A BACK CHARACTER
          D2   13     06F6  1613          BEQL    930$                     ; YES THEN DISPATCH
   51     06   9A     06F8  1614          MOVZBL  #TTY$K_ET_FORWARD_CHAR,R1; GO FOWARD A CHARACTER
53    43 8F   91     06FB  1615          CMPB    #^A/C/,R3                ; IS IT A FORWARD
          C9   13     06FF  1616          BEQL    930$                     ; HANDLE IT
   51     0B   9A     0701  1617          MOVZBL  #TTY$K_ET_RECALL,R1      ; RECALL A COMMAND
53    41 8F   91     0704  1618          CMPB    #^A/A/,R3                ; IS THIS RECALL?
          C0   13     0708  1619          BEQL    930$
06 20 A4    10   E0     070A  1620          BBS     #TRM$V_TM_NORECALL,TTY$L_RB_MOD(R4),190$; CHECK NORE_ALL
53    42 8F   91     070F  1621          CMPB    #^A/B/,R3                ; IS THIS A DOWN ARROW?
          B5   13     0713  1622          BEQL    930$                     ; YES THEN DROP IT
                     0715  1623 ;
                     0715  1624 ; DON'T TERMINATE ON THIS SEQUENCE UNLESS HE REALY WANTS IT
                     0715  1625 ; (THE MODE OF THE READ IS ESCAPE
                     0715  1626 ;
                     0715  1627 190$:   IF_STATE ESCAPE,199$             ; IF THIS IS AN ESCAPE SEQUENCE READ THEN
                     0719  1628 970$:                                    ; TERMINATE ON ESCAPE SEQUENCES.
51   58 A5   D0     0719  1629          MOVL    UCB$L_IRP(R5),R1         ; MAKE SURE WE HAVE THE IRP ADDRESS
     38 A1   D4     071D  1630          CLRL    IRP$L_MEDIA(R1)          ; NO THEN CLEAN THE SEQUENCE OFF THE TERMINA
      F8E2   31     0720  1631          BRW     GETNEXTCHAR              ; AND GET THE NEXT CHARACTER
                     0723  1632 ;
                     0723  1633 ; NORMAL ESCAPE SEQUENCE TERMINATION
                     0723  1634 ;
                     0723  1635
   53    30   91     0723  1636 198$:   CMPB    #^A/0/,R3                ; IS THIS A NUMBER
          10   14     0726  1637          BGTR    900$                     ; NO THEN EXIT
   53    39   91     0728  1638          CMPB    #^A/9/,R3                ; ...
          0B   19     072B  1639          BLSS    900$                     ;
48 A4    0A   A4     072D  1640          MULW    #10,TTY$W_RB_ESCTKN(R4)  ; SHIFT THE NUMBER OVER
   53    30   A2     0731  1641          SUBW    #^A/0/,R3                ; SUBTRACT OUT 0
48 A4    53   A0     0734  1642          ADDW    R3,TTY$W_RB_ESCTKN(R4)   ;ADD IT IN
      FE58   31     0738  1643 900$:     BRW     210$
```

```
                     073B  1645 ;
                     073B  1646 ; SEE IF DELETE SEQUENCE SHOULD BE TERMINATED
                     073B  1647 ;
7F 8F    53    91    073B  1648 INDELST:CMPB    R3,#TTY$C_DELETE        ; IS THIS CHARACTER A DELETE?
         0E    13    073F  1649          BEQL    50$                     ; IF YES THEN GO ON
      0C A4    B6    0741  1650          INCW    TTY$W_TA_INAHD(R4)      ; NO THEN PUT THE CHARACTER BACK
                     0744  1651          CLR_STATE DEL                   ; TERMINATE DELETE SEQUENCE
                     0748  1652 ;
                     0748  1653 ; TERMINATE DELETE SEQUENCE IF NOT IN NO-ECHO
                     0748  1654 ;
53    5C 8F    9A    0748  1655          MOVZBL  #^A/\/,R3               ; SET OUTPUT CHARACTER
       FA38    31    074C  1656          BRW     FORMAT_CHAR             ; FORMAT THE CHARACTER
                     074F  1657
       FDF6    31    074F  1658 50$:     BRW     40$
                     0752  1659
```

```
                                0752  1661 ;
                                0752  1662 ; OPEN THE BUFFER
                                0752  1663 ; OR JUST OVERSTRIKE
                                0752  1664 ;
                                0752  1665 OPEN:
                                0752  1666 ;
                                0752  1667 ; SPREAD THE BUFFER TO PUT THIS CHARCTER IN PLACE
                                0752  1668 ;
                          3F    BB  0752  1669          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; SAVE THE REGISTERS OVER THE MOVE
                          51    D4  0754  1670          CLRL    R1                      ; SET FORWARD MODE
                       07F9     30  0756  1671          BSBW    CHAR_SIZE               ; GET THE CHARACTER SIZE
                                0759  1672 ;
                                0759  1673 ; IF NOT AT THE END OF LINE THEN DON'T LET THE USER WRAP THE LINE
                                0759  1674 ;
                                0759  1675          IF_STATE <WRAP,TABRIGHT>,194$           ;...
 38 A4    00FC C5    53      A1  0763  1676 169$:    ADDW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); AND ADD IT INTO THE CURR
            51    30 A4      3C  076A  1677          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE LOCATION OF THE LAST CHARACTER
       14 A4     2C B441     9E  076E  1678          MOVAB   @TTY$L_RB_LIN(R4)[R1],TTY$L_RB_ECHSTR(R4); GET THE ADDRESS
          44 A4     01      B0  0774  1679          MOVW    #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); AND SETUP THE ECHO MODE
                                0778  1680          IF_STATE NOECHO,193$            ; DON'T ECHO IF NOT ECHOING...
                                077C  1681          SET_STATE EDITREAD              ; AND SET THE EDIT READ STATE
                                0780  1682 193$:    IF_NOT_STATE OVERSTRIKE,158$    ; NOT IN OVERSTRIKE THEN HANDLE NORMALY
             09    0C AE      91  0784  1683          CMPB    12(SP),#TTY$C_TAB       ; WAS THIS CHARACTER A TAB?
                     2D      13  0788  1684          BEQL    158$                    ; YES THEN INSERT IT.
                                078A  1685
                32 A4        B7  078A  1686          DECW    TTY$W_RB_LINREST(R4)    ; IN OVERSTRIKE WE DELETE WHILE TYPING
                3C A4        B7  078D  1687          DECW    TTY$W_RB_TXTOFF(R4)     ; FAKE OUT THE REST OF THE LOGIC
                                0790  1688
                01    53     D1  0790  1689          CMPL    R3,#1                   ; IS THIS A SHORT ECHOING CHARACTER
                        2C     12  0793  1690          BNEQ    168$                    ; NO THEN OUTPUT THE WHOLE THING
             09    14 B4      91  0795  1691          CMPB    @TTY$L_RB_ECHSTR(R4),#TTY$C_TAB; ARE WE ON TOP OF A TAB?
                        0C     13  0799  1692          BEQL    157$                    ; YES THEN HANDLE SPECAILLY
                44 A4        B4  079B  1693          CLRW    TTY$W_RB_MODE(R4)       ; CLEAN OUT THE MODE FOR LATER
                14 A4        D4  079E  1694          CLRL    TTY$L_RB_ECHSTR(R4)     ; MAKE SURE THE STRING IS ZERO
                                07A1  1695          CLR_STATE EDITREAD              ; AND DON'T DO THE EDITREAD ECHO (ONLY
                                07A5  1696                                          ; ONE CHARACTER NO NEED)
                        1A     11  07A5  1697          BRB     168$                    ; AND DON'T BOTHER WITH THE MOVC
 53    38 A4     03    00      EF  07A7  1698 157$:    EXTZV   #0,#3,TTY$W_RB_CPZCUR(R4),R3; GET THE LENGTH OF THE TAB
                        53     D5  07AD  1699          TSTL    R3                      ; DELETE THE TAB WHEN IT EXPANDS TO ZERO
                        1B     13  07AF  1700          BEQL    171$                    ; MAKE IT DISAPPEAR
                32 A4        B6  07B1  1701          INCW    TTY$W_RB_LINREST(R4)    ; ELSE JUST INSERT THE CHARACTER
                3C A4        B6  07B4  1702          INCW    TTY$W_RB_TXTOFF(R4)
                   51        D6  07B7  1703 158$:    INCL    R1                      ; INCREMENT THE REGISTER BY 1 TO GET THE
                                07B9  1704                                          ; NEW POSITION
                32 A4        28  07B9  1705          MOVC3   TTY$W_RB_LINREST(R4),-  ; AND MOVE THE CHARACTERS OVER
          2C B441    14 B4      07BC  1706                  @TTY$L_RB_ECHSTR(R4),@TTY$L_RB_LIN(R4)[R1]
                   3F     BA  07C1  1707 168$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE THE REGISTERS
    0B A4     01    32 A4      81  07C3  1708          ADDB3   TTY$W_RB_LINREST(R4),#1,TTY$B_RB_ECHLEN(R4);AND LENGTH OF STRING TO
                   FDBC     31  07C9  1709          BRW     159$                    ; AND INSERT THE CHARACTER
                30 A4        B6  07CC  1710 171$:    INCW    TTY$W_RB_LINOFF(R4)     ; LOOK FROM THIS CHARACTER ON
                 088B        30  07CF  1711          BSBW    TABRIGHT                ; MAKE SURE THE TAB AFTER THIS POSITION
                                07D2  1712                                          ; FLAG IS SET CORRECTLY.
                30 A4        B7  07D2  1713          DECW    TTY$W_RB_LINOFF(R4)     ; KEEP LINOFF THE SAME AS WE STARTED WITH
                   EA        11  07D5  1714          BRB     168$                    ; JOIN THE FLOW
                                07D7  1715 ;
                                07D7  1716 ; ONCE WRAP IS DETERMINED THEN CHECK SPECIFICALLY IF THIS CHARACTER IS VALID
                                07D7  1717 ;
```

```
                07D7  1718 194$:    IF_NOT_STATE OVERSTRIKE,234$     ; DON'T EVER INSERT AFTER THE WRAP
   01    53    B1  07DB  1719         CMPW    R3,#1                   ; IS THIS A SINGLE CHARACTER EXPANSION
         06    12  07DE  1720         BNEQ    234$                    ; NO THEN DON'T INSERT IT
09    0C AE    91  07E0  1721         CMPB    12(SP),#TTY$C_TAB       ; WAS THIS CHARACTER A TAB?
         05    12  07E4  1722         BNEQ    235$                    ; NO THEN OVERSTRIKE IT
         3F    BA  07E6  1723 234$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE THE REGISTERS
       F81A    31  07E8  1724         BRW     GETNEXTCHAR             ; AND GET THE NEXT CHARACTER
       FF75    31  07EB  1725 235$:    BRW     169$
```

```
                        07EE  1727 ;
                        07EE  1728 ; QUOTING  CHARACTER DETECTED
                        07EE  1729 ;
 0D 04 A2    14   E4    07EE  1730 QUOTE:  BBSC    #TTY$V_ST_EDITING,4(R2),65$; IF EDITING IS SET THEN CLEAR IT
                        07F3  1731         IF_STATE       ESC,65$          ; IF ESCAPE THEN DON'T TURN OFF EDITING YET
                        07F7  1732         SET_STATE EDITING               ; IF IT NOT EDITING THEN SET EDITING
                        07FB  1733         CLR_STATE QUOTING               ; AND CLEAR QUOTING
           FD60   31    0800  1734 65$:    BRW     67$                     ; RETURN TO THE NORMAL PLACE
```

```
                     0803  1736          .SBTTL  Specail input character dispatcher
                     0803  1737 ;++
                     0803  1738 ;
                     0803  1739 ; SERVICE ROUTINES TO MOVE READATA
                     0803  1740 ;
                     0803  1741 ; SPECAIL EDITING CHARACTER HANDELING
                     0803  1742 ;
                     0803  1743 EDITINGCHAR:
                     0803  1744          IF_STATE <EDITING>,70$          ; EDITING THEN NO RANGE CHECK NECESSARY
 04   51   D1       0807  1745          CMPL    R1,#TTY$K_EDITNORMAL    ; NOT EDITING THEN ANYTHING ADVANCED
      27   14       080A  1746          BGTR    162$                    ; IS BAD
                     080C  1747 70$:     CASE    R1,TYPE=L,<-
                     080C  1748                  151$,-                  ; IS NORMAL CHARACTER (SHOULD NEVER BE USED)
                     080C  1749                  CTRLU,-                 ; IS CONTROL-U
                     080C  1750                  CTRLR,-                 ; INDICATES CONTROL-R
                     080C  1751                  DELCHAR,-               ; INDICATES DELETE CHARCTER
                     080C  1752                  ESCAPE_CHAR,-           ; INDICATES ESCAPE CHARCTER
                     080C  1753                  BACK_CHAR,-             ; MOVE BACK ONE CHARACTER
                     080C  1754                  FORWARD_CHAR,-          ; MOVE FOWARD ONE CHARACTER
                     080C  1755                  MOVE_EOL,-              ; MOVE TO THE END OF LINE
                     080C  1756                  MOVE_BOL,-              ; MOVE TO THE BEGINNING OF LINE
                     080C  1757                  DELETE_WORD,-           ; DELETE WORD TO THE LEFT
                     080C  1758                  QUOTING,-               ; QUOTE CHARACTER
                     080C  1759                  RECALL,-                ; RECALL THE LAST COMMAND
                     080C  1760                  TOGGELINSOV-            ; TOGGEL INSERT/OVERSTRIKE MODE
                     080C  1761                  >
                     082A  1762          IF_STATE TERMNORM,162$          ; TERMINATE ON USUAL
                     082E  1763                                          ; CHARACTERS
 0E   51   D1       082E  1764          CMPL    R1,#TTY$K_ET_TERMINATE  ; WAS THIS A REAL TERMINATOR CHARACTER
      1E   12       0831  1765          BNEQ    165$                    ; NO THEN DON'T LET IT THRU
                     0833  1766 162$:    IF_STATE        <ESC>,166$      ; IF IN AN ESCAPE SEQUENCE THEN LET IT THRU
0F 1C B4   53   E0   0837  1767          BBS     R3,@TTY$L_RB_TERM(R4),164$; IS THIS CHARACTER A TERMINATOR
 0D   53   91       083C  1768          CMPB    R3,#TTY$C_CR            ; WE ARE PREPARED TO HANDLE <CR>
      13   13       083F  1769          BEQL    800$                    ; SO LET IT THRU ANYWAY
                     0841  1770          IF_STATE <QUOTING,EDITING>,165$ ; NO THEN DROP IT IF WE ARE EDITING
    FD22   31       0848  1771 167$:    BRW     151$
    FD6E   31       084B  1772 164$:    BRW     TERMFOUND               ; TERMINATOR DETECTED
    FE0E   31       084E  1773 166$:    BRW     ESCINPROG               ; ESCAPE SEQUENCE INPROGRESS
    F7B1   31       0851  1774 165$:    BRW     GETNEXTCHAR             ; GET THE NEXT CHAR
  32 A4    B5       0854  1775 800$:    TSTW    TTY$W_RB_LINREST(R4)    ; ARE WE AT THE END OF THE LINE
     EF    13       0857  1776          BEQL    167$                    ; YES THEN CR IS LEGAL
    F7A9   31       0859  1777          BRW     GETNEXTCHAR
                     085C  1778
                     085C  1779          .disable lsb
                     085C  1780
```

```
                        085C  1782 ;++
                        085C  1783 ; RECALLING
                        085C  1784 ;
                        085C  1785 ; DESCRIPTION:
                        085C  1786 ;       THIS STATE IS USED WHEN THE LAST COMMAND TYPED IS REQUESTED.
                        085C  1787 ; THIS ROUTINE WILL REMOVE CHARACTERS FROM THE RECALL BUFFER THEN
                        085C  1788 ; ALLOW THE NORMAL INSERT CHARACTER LOGIC TAKE AFFECT.  THE CHARACTERS
                        085C  1789 ; IN THE RECALL BUFFER ARE PROCESSED AS IF THEY WERE NO FORMAT.
                        085C  1790 ;
                        085C  1791 ; INPUTS:
                        085C  1792 ;       R5 = UCB ADDRESS
                        085C  1793 ;
                        085C  1794 ; OUTPUTS:
                        085C  1795 ;       SEE MOVEREADATA
                        085C  1796 ;--
                        085C  1797 RECALLING:
 51   00E4 C5    D0      085C  1798         MOVL    UCB$L_TT_TYPAHD(R5),R1  ; GET THE ADDRESS OF THE TYPEAHEAD BUFFER
            25    13      0861  1799         BEQL    20$                     ; NONE THEN LET MOVEREADATA HANDLE IT
   54    78 A5    D0      0863  1800         MOVL    UCB$L_SVAPTE(R5),R4     ; SETUP THE READ PACKET ADDRESS
   53    0E A1    3C      0867  1801         MOVZWL  TTY$W_TA_RCLOFF(R1),R3  ; GET THE ADDRESS OF THE NEXT CARACTER
 53   18 A143    9A      086B  1802         MOVZBL  TTY$A_TA_RCL(R1)[R3],R3 ; GET THE CHARACTER
         0E A1    B6      0870  1803         INCW    TTY$W_TA_RCLOFF(R1)     ; MOVE ON TO THE NEXT CHARACTER
14 A1    0E A1    B1      0873  1804         CMPW    TTY$W_TA_RCLOFF(R1),TTY$W_TA_RCLSIZ(R1); ARE WE FINISHED
            07    18      0878  1805         BGEQ    5$                      ; YES THEN TURN OFF RECALL AND CONTINUE
40 A4    0E A1    B1      087A  1806         CMPW    TTY$W_TA_RCLOFF(R1),TTY$W_RB_TXTSIZ(R4); ARE WE GOING TO OVERFLOW
            04    19      087F  1807         BLSS    10$                     ; THE USERS BUFFER
                          0881  1808 5$:     CLR_STATE RECALL                ; YES THEN MAKE THIS THE LAST TIME
          FCE9    31      0885  1809 10$:    BRW     INSERT_CHAR             ; AND INSERT THE CHARACTER
          FC9B    31      0888  1810 20$:    BRW     MOVEREADATA             ; ABORT EXIT
```

```
                                088B  1812          .SBTTL  Post typeahead character action routines
                                088B  1813
                                088B  1814 ;++
                                088B  1815 ; BACK_CHAR
                                088B  1816 ;       Move backward a single character.
                                088B  1817 ;--
                                088B  1818 BACK_CHAR:
                    3A A4   B5  088B  1819          TSTW    TTY$W_RB_CPZORG(R4)     ; START AT ZERO?
                      03    13  088E  1820          BEQL    10$                     ; YES THEN NO SPECAIL ACTION
                    0885    30  0890  1821          BSBW    ZERO_CPZORG             ; NO THEN MAKE SURE WE DO
                    30 A4   B7  0893  1822 10$:     DECW    TTY$W_RB_LINOFF(R4)     ; BACKUP A CHARACTER
                      2A    19  0896  1823          BLSS    50$                     ; AT THE END THEN DON'T ALLOW IT
                    32 A4   B6  0898  1824          INCW    TTY$W_RB_LINREST(R4)    ; UPDATE THE REST POINTER
                51  30 A4   3C  089B  1825          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET TO THE CHARACTER
              53  2C B441   9A  089F  1826          MOVZBL  @TTY$C_RB_LIN(R4)[R1],R3; GET THE CHARCTER
                  53  09    91  08A4  1827          CMPB    #TTY$C_TAB,R3           ; IS THIS A TAB
                      05    12  08A7  1828          BNEQ    20$                     ; NO THEN CONTINUE
                                08A9  1829          SET_STATE TABRIGHT              ; ELSE MARK THIS INFORMATION
                  51  01    CE  08AE  1830 20$:     MNEGL   #1,R1                   ; GOING BACKWARDS TO
                    069E    30  08B1  1831          BSBW    CHAR_SIZE               ; CHECK THE SIZE OF THE CHARACTER
        38 A4   00FC C5  53   A3  08B4  1832          SUBW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURSOR POSITI
                                08BB  1833          IF_STATE NOECHO,60$
                    07E7    31  08BF  1834          BRW     UPDATE_CURSOR           ; SET THE CURSOR TO IT'S NEW LOCATION
                                08C2  1835
                    30 A4   B4  08C2  1836 50$:     CLRW    TTY$W_RB_LINOFF(R4)     ; RESET THE POINTER
                    F9D1    31  08C5  1837 60$:     BRW     DROP                    ; AND DROP THE CHARACTER
```

```
                                                J 8
                     08C8  1839          .enable lsb
                     08C8  1840 ;
                     08C8  1841 ; CONTROL R PROCESSING
                     08C8  1842 ;
                     08C8  1843 CTRLR:   IF_STATE <ESC>,150$                 ; NO ACTION IF ESCAPE OR NO ECHO
                     08CC  1844          IF_NOT_STATE NOECHO,100$            ; BRANCH IF ECHO
           00   E1   08D0  1845          BBC     #TT2$V_LOCALECHO,-
      2D 48 A5        08D2  1846                  UCB$L_DEVDEPND2(R5),150$; BRANCH IF NOT LOCAL ECHO
                     08D5  1847
   54  0000'CF   9E   08D5  1848 100$:   MOVAB   W^TTY$A_CTRLR,R4           ; ADDRESS MULTIECHO STRING
           0C   E1   08DA  1849 110$:   BBC     #TT2$V_EDITING,-
      1C 48 A5        08DC  1850                  UCB$L_DEVDEPND2(R5),130$        ; NO THEN USE NORMAL ECHO STRING
17 44 A5    0C   E1   08DF  1851          BBC     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),130$; DON'T SKIP STUFF
                     08E4  1852                                              ; ON A HARD COPY TERMINAL
   51    78 A5   D0   08E4  1853          MOVL    UCB$L_SVAPTE(R5),R1        ; GET THE PACKET ADDRESS
         3A A1   B5   08E8  1854          TSTW    TTY$W_RB_CPZORG(R1)        ; WERE WE AT THE FIRST COLUMN WHEN
            04   12   08EB  1855          BNEQ    120$                       ; WE STARTED THE READ
                     08ED  1856          SET_STATE <SKIPCRLF>                ; NO LINEFEEDS AFTER <CR>
           18   E1   08F1  1857 120$:   BBC     #TT2$V_ANSICRT,-           ; IS IT AN ANSI CRT
      05 48 A5        08F3  1858                  UCB$L_DEVDEPND2(R5),130$; NO THEN USE NORMAL ECHO STRING
   54  0000'CF   9E   08F6  1859          MOVAB   W^TTY$A_ANSI_DEOL,R4       ; YES THEN WE CAN USE CRT MODE
                     08FB  1860 130$:
                     08FB  1861          SET_STATE       <EDITREAD>          ; SET CONTROL R STATE
         075E   31   08FF  1862          BRW     STRTMULTI                  ; START MULTIPLE OUTPUT SEQUENCE
                     0902  1863
         FC21   31   0902  1864 150$:   BRW     MOVEREADATA
```

```
                                0905  1866 ;
                                0905  1867 ; CONTROL U PROCESSING
                                0905  1868 ;
                         0239   30  0905  1869 10$:    BSBW    DELESCAPE                       ; DELETE THE ESCAPE SEQUENCE
                                0908  1870 CTRLU:    IF_STATE ESC,10$                        ; IF IN ESCAPE SEQUENCE THEN DELETE ESCAPE
                           30 A4  B5  090C  1871        TSTW    TTY$W_RB_LINOFF(R4)             ; ANY DATA TO CONTROL-U?
                              69  13  090F  1872        BEQL    50$                             ; NO THEN DON'T DO THE WORK
                     3C A4    30 A4  A2  0911  1873        SUBW    TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4) ; CLEAN OUT THE TEXT OFFSETO
                           0B A4  94  0916  1874        CLRB    TTY$B_RB_ECHLEN(R4)             ; CLEAN THE EXTRA
                           38 A4  B4  0919  1875        CLRW    TTY$W_RB_CPZCUR(R4)             ; MOVE TO THE BEGINNING OF THE LINE
                           3A A4  B5  091C  1876        TSTW    TTY$W_RB_CPZORG(R4)             ; WAS THIS AT THE RIGHT
                              04  12  091F  1877        BNEQ    12$                             ; NO THEN USE THE OLD METHOD
                        44 A4    06  9B  0921  1878        MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); EXIT WHEN NOT DOING ANYTHING EL
                           32 A4  B5  0925  1879 12$:    TSTW    TTY$W_RB_CINREST(R4)            ; IS THERE A REST OF THE LINE
                              26  13  0928  1880        BEQL    13$                             ; NO THEN HANDLE NORMALLY
                              3F  BB  092A  1881        PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; SAVE THE REGISTERS
                        51    30 A4  3C  092C  1882        MOVZWL  TTY$W_RB_LINOFF(R4),R1          ; GET THE NUMBER OF CHARACTERS
                  2C B4 2C B441    32 A4  28  0930  1883        MOVC3   TTY$W_RB_LINREST(R4),@TTY$L_RB_LIN(R4)[R1],@TTY$L_RB_LIN(R4);
                              3F  BA  0938  1884        POPR    #^M<R0,R1,R2,R3,R4,R5>          ; AND RESTORE THE REGISTERS
                           3A A4  B5  093A  1885        TSTW    TTY$W_RB_CPZORG(R4)             ; WAS THIS AT THE RIGHT
                              1B  12  093D  1886        BNEQ    14$                             ; NO THEN USE THE OLD METHOD
                        44 A4    01  9B  093F  1887        MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); SETUP THE NEW ECHO MODE
                     14 A4    2C A4  D0  0943  1888        MOVL    TTY$L_RB_CIN(R4),TTY$L_RB_ECHSTR(R4); THE LOCATION OF THE FIRST CHAR
                     0B A4    32 A4  90  0948  1889        MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH
                           38 A4  B4  094D  1890        CLRW    TTY$W_RB_CPZCUR(R4)             ; AND GO ALL THE WAY TO THE BEGINNING
                        64    2C A4  D1  0950  1891 13$:    CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE
                              04  12  0954  1892        BNEQU   14$                             ;NO THEN CONTINUE AS USUAL
                        44 A4    07  9B  0956  1893        MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); ECHO THE PROMPT TOO
                           30 A4  B4  095A  1894 14$:    CLRW    TTY$W_RB_CINOFF(R4)             ; NOW ZERO THE LINE OFFFSET
                                095D  1895
                                095D  1896        IF_STATE NOECHO,15$                     ; BRANCH IF ECHO
                     54 0000'CF    9E  0961  1897        MOVAB   W^TTY$A_CTRLU,R4                ; ADDRESS MULTIECHO STRING
                           FF71    31  0966  1898        BRW     110$                            ; CONTINUE
                                0969  1899
                           FBBA    31  0969  1900 15$:    BRW     MOVEREADATA                     ;
                                096C  1901 55$:    CLR_STATE <WRAP,CURSOR>
                           FF91    31  0974  1902        BRW     CTRLU
                           FF4E    31  0977  1903 57$:    BRW     CTRLR
                                097A  1904 ;
                                097A  1905 ; NO DATA TO CONTROL-U
                                097A  1906 ;
                           32 A4    B5  097A  1907 50$:    TSTW    TTY$W_RB_LINREST(R4)            ; ANY DATA AFTER THIS POINT
                              EA    12  097D  1908        BNEQ    15$                             ; YES THEN JUST RETURN
                           3C A4    B5  097F  1909        TSTW    TTY$W_RB_TXTOFF(R4)             ; IS THERE ANY REASON TO DO THIS
                              E5    13  0982  1910        BEQL    15$                             ; NO THEN DON'T BOTHER
                           3C A4    B7  0984  1911        DECW    TTY$W_RB_TXTOFF(R4)             ; REMOVE THE LAST CHARACTER
                            0637    30  0987  1912        BSBW    FIND_BOL                        ; THEN FIND THE BEGINNING
                                098A  1913                                                    ; OF THE PREVIOUS LINE
                           3C A4    B5  098A  1914        TSTW    TTY$W_RB_TXTOFF(R4)             ; IS THERE ANY MORE DATA LEFT
                              E8    13  098D  1915        BEQL    57$                             ; NO THEN JUST MAKE SURE THE PROMPT
                                098F  1916                                                    ; ECHO'S
                  D8 48 A5    18    E1  098F  1917        BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),55$; REECHO ON NON ANSI TERMINALS
                           3A A4    B5  0994  1918        TSTW    TTY$W_RB_CPZORG(R4)             ; WAS THIS AT THE RIGHT
                              D3    12  0997  1919        BNEQ    55$                             ; NO THEN USE THE OLD METHOD
                     3C A4    30 A4  A2  0999  1920        SUBW    TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4); DELETE THE LINE
                           30 A4    B4  099E  1921        CLRW    TTY$W_RB_LINOFF(R4)             ; ...
                        64    2C A4    D1  09A1  1922        CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); DOES A PROMPT NEED TO BE ECHOED
```

```
                    0B   12  09A5  1923          BNEQ    60$                             ; NO THEN DON'T ECHO A PROMPT
              44 A4 07   9B  09A7  1924          MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SET UP TO ECHO THE PROMPT
                 0B A4   94  09AB  1925          CLRB    TTY$B_RB_ECHLEN(R4)             ; NO STRING TO OUTPUT
                             09AE  1926          SET_STATE EDITREAD                      ; TELL THE REST OF THE SYSTEM ABOUT THE PROM
                             09B2  1927
                 38 A4   B4  09B2  1928 60$:     CLRW    TTY$W_RB_CPZCUR(R4)             ; CLEAR THE CURSOR POSITION
                             09B5  1929          SET_STATE SKIPCRLF                      ; SKIP THE FREE LINEFEED
           54 00000000'EF    9E  09B9  1930          MOVAB   TTY$A_ANSI_UPCEL,R4         ; GET THE STRING ADDRESS
                   069D    31  09C0  1931          BRW     STRTMOLTI                       ; AND DELETE THIS LINE
                             09C3  1932          .disable lsb
```

```
                        09C3  1934
                        09C3  1935 ;++
                        09C3  1936 ; DELCHAR - DELETE CHARACTER ROUTINE
                        09C3  1937 ;
                        09C3  1938 ; FUNCTIONAL DESCRIPTION:
                        09C3  1939 ;
                        09C3  1940 ; THIS ROUTINE DELETES THE LAST TYPED CHARACTER FROM THE READ BUFFER.
                        09C3  1941 ; THEN IT SETS UP THE PROPER ECHO SEQUENCE FOR THE DELETED CHARACTER.
                        09C3  1942 ;
                        09C3  1943 ; INPUTS:
                        09C3  1944 ;
                        09C3  1945 ;       R2 = ADDRESS OF TTY STATE VECTOR
                        09C3  1946 ;       R5 = UCB ADDRESS
                        09C3  1947 ;
                        09C3  1948 ; OUTPUTS:
                        09C3  1949 ;
                        09C3  1950 ;       R2 = ADDRESS OF TTY STATE VECTOR
                        09C3  1951 ;       R5 = UCB ADDRESS
                        09C3  1952 ;--
                        09C3  1953 DELCHAR:                                     ;
          54   78 A5   D0  09C3  1954         MOVL    UCB$L_SVAPTE(R5),R4       ; ADDRESS READ BUFFER BLOCK
                        09C7  1955 ;
                        09C7  1956 ; IF ESCAPE SEQUENCE IN PROGRESS THEN DELETE ENTIRE STRING
                        09C7  1957 ;
                        09C7  1958         IF_STATE ESC,25$                     ; DELETE ESCAPE SEQUENCE
            30 A4   B7  09CB  1959         DECW    TTY$W_RB_LINOFF(R4)          ; ANY DATA
              30    19  09CE  1960         BLSS    28$                          ; IF EQL THEN NO
        51    30 A4   3C  09D0  1961         MOVZWL  TTY$W_RB_LINOFF(R4),R1       ; GET THE POINTER
    53  2C B441   9A  09D4  1962         MOVZBL  @TTY$[_RB_LIN(R4)[R1],R3; GET THE CHARACTER
        53    0D  91  09D9  1963         CMPB    #TTY$C_CR,R3                 ; IS THIS CHARACTER A RETURN
              22  13  09DC  1964         BEQL    28$                          ; YES THEN ADJUST THE LINE
            3C A4   B7  09DE  1965         DECW    TTY$W_RB_TXTOFF(R4)          ; ADJUST POINTER
                        09E1  1966         IF_NOT_STATE NOECHO,5$               ; BRANCH IF ECHO
              00    E1  09E5  1967         BBC     #TT2$V_LOCALECHO,-
          6F 48 A5        09E7  1968                 UCB$L_DEVDEPND2(R5),30$ ; BRANCH IF NOT LOCAL ECHO
                        09EA  1969
                        09EA  1970 ;
                        09EA  1971 ; TEST FOR SPECIAL DELETES
                        09EA  1972 ;
            09    53    91  09EA  1973 5$:     CMPB    R3,#TTY$C_TAB                ; TAB?
                  03    13  09ED  1974         BEQL    8$                           ; IF NEQ THEN OUTPUT JUST THE CHARACTER
                00A3    31  09EF  1975         BRW     55$
    65 44 A5      0C    E0  09F2  1976 8$:     BBS     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),40$; BR IF SCOPE
                00DB    31  09F7  1977         BRW     75$                          ; BR IF NOT SCOPE
                        09FA  1978
                  0144  30  09FA  1979 25$:    BSBW    DELESCAPE                    ; DELETE ESCAPE SEQUENCE
                  FB26  31  09FD  1980         BRW     MOVEREADATA                  ; AND GO BACK TO WORK
                        0A00  1981
                30 A4   B4  0A00  1982 28$:    CLRW    TTY$W_RB_LINOFF(R4)          ; MAKE THE COUNT ZERO
                32 A4   B5  0A03  1983         TSTW    TTY$W_RB_LINREST(R4)         ; is there nothing left?
                   51   12  0A06  1984         BNEQ    30$                          ; NO THEN JUST RETURN
                3C A4   B5  0A08  1985         TSTW    TTY$W_RB_TXTOFF(R4)          ; NO LINE WAS WRAPPED
                   4C   13  0A0B  1986         BEQL    30$                          ; IF THIS COUNT IS ZERO
                3C A4   B7  0A0D  1987         DECW    TTY$W_RB_TXTOFF(R4)          ; REMOVE THE LAST CHARACTER
                 05AE   30  0A10  1988         BSBW    FIND_BOL                     ; YES THEN CALCULATE THE NEW BEGINNING
                        0A13  1989                                                  ; OF LINE
    33 48 A5      18    E1  0A13  1990         BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),120$; REECHO ON NON ANSI TERMINAL
```

```
                3A A4   B5   0A18  1991          TSTW    TTY$W_RB_CPZORG(R4)      ; WAS THIS AT THE RIGHT
                   2E   12   0A1B  1992          BNEQ    120$                     ; NO THEN USE THE OLD METHOD
            44 A4   01   9B   0A1D  1993          MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); ECHO THE DATA
         0B A4   30 A4   90   0A21  1994          MOVB    TTY$W_RB_CINOFF(R4),TTY$B_RB_ECHLEN(R4); AND THE STRING TO OUTPUT
         14 A4   2C A4   D0   0A26  1995          MOVL    TTY$L_RB_LIN(R4),TTY$L_RB_ECRSTR(R4); AND THE ADDRESS
            64   2C A4   D1   0A2B  1996          CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); DOES A PROMPT NEED TO BE ECHOED
                   04   12   0A2F  1997          BNEQ    110$                     ; NO THEN DON'T ECHO A PROMPT
            44 A4   07   9B   0A31  1998          MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SET UP TO ECHO THE PROMPT
                        0A35  1999
                        0A35  2000 110$:   SET_STATE <SKIPCRLF,EDITREAD>    ; SKIP THE FREE LINEFEED
            38 A4   02   AE   0A3D  2001          MNEGW   #2,TTY$W_RB_CPZCUR(R4)   ; DON'T RESET CURSOR
      54    00000000'EF   9E   0A41  2002          MOVAB   TTY$A_ANSI_DPCEL,R4      ; GET THE STRING ADDRESS
               0615   31   0A48  2003          BRW     STRTMULTI                ; AND DELETE THIS LINE
                        0A4B  2004
                        0A4B  2005 120$:   SET_STATE <EDITREAD>             ; RESTORE THE LINE TO IT'S FULL VIGOR
      54    00000000'EF   9E   0A4F  2006          MOVAB   TTY$A_CTRLR,R4           ; GET A NEW LINE
               0607   31   0A56  2007          BRW     STRTMULTI                ; and return to the main loop
                        0A59  2008
               FACA   31   0A59  2009 30$:    BRW     MOVEREADATA              ; VECTOR TO CONTINUE
                        0A5C  2010
                        0A5C  2011 ;
                        0A5C  2012 ; DELETE A TAB ON A SCOPE
                        0A5C  2013 ;
            00FC C5   B5   0A5C  2014 40$:    TSTW    UCB$W_TT_CURSOR(R5)      ; Have we wrapped?
                  47   13   0A60  2015          BEQL    60$                      ; Branch to simulate CTRL-R, if yes.
                048A   30   0A62  2016          BSBW    BACK_TAB                 ; GET THE TAB LENGTH
      42 A5   00FC C5   B1   0A65  2017          CMPW    UCB$W_TT_CURSOR(R5), UCB$W_DEVBUFSIZ(R5) ; At extreme right?
                  11   1F   0A6B  2018          BLSSU   45$                      ; Branch if within screen bounds.
                  1C   1A   0A6D  2019          BGTRU   47$                      ; Branch if beyond right edge.
                  53   D6   0A6F  2020          INCL    R3                       ; Adjust number of backspaces for
       53   FFFFFFF8 8F   CA   0A71  2021          BICL    #^C7, R3                 ; right-hand edge effects.
                  41   13   0A78  2022          BEQL    66$                      ; If no backspace, just delete at edge.
               00FC C5   B7   0A7A  2023          DECW    UCB$W_TT_CURSOR(R5)      ; Adjust cursor value for one less BS.
   00DC C5     07   53   A3   0A7E  2024 45$:    SUBW3   R3,#7,UCB$W_TT_MULTILEN(R5); GET THE LENGTH CORRECTLY
      54    0001'CF   9E   0A84  2025          MOVAB   W^TTY$A_DELCRTTAB+1,R4; Get address backspaces string.
                  45   11   0A89  2026          BRB     70$                      ; Go output them.
                        0A8B  2027
               53   07   A2   0A8B  2028 47$:    SUBW    #7, R3                   ; Adjust off-the-screen cursor
            00FC C5   53   A0   0A8E  2029          ADDW    R3, UCB$W_TT_CURSOR(R5) ; position for deleted tab.
                  26   11   0A93  2030          BRB     66$                      ; Go do delete-at-edge.
                        0A95  2031
                        0A95  2032 ;
                        0A95  2033 ; NORMAL CHARACTER DELETE
                        0A95  2034 ;
                        0A95  2035 55$:
                  F0 8F   93   0A95  2036          BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
               0000'CF43         0A98  2037          W^TTY$A_TYPE[R3]                 ; TEST FOR SPECIAL
                  14   12   0A9C  2038          BNEQ    62$                      ; NON PRINTING CHARACTER
                        0A9E  2039
            32 44 A5   0C   E1   0A9E  2040          BBC     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),75$; BR IF NOT SCOPE
                        0AA3  2041 ;
                        0AA3  2042 ; IF THE CURSOR IS AT THE LEFT MARGIN AND DATA IS PRESENT,
                        0AA3  2043 ; FORCE A CONTROL R IN STEAD OF A BACKSPACE.
                        0AA3  2044 ;
            00FC C5   B5   0AA3  2045 59$:    TSTW    UCB$W_TT_CURSOR(R5)      ; CURSOR AT LEFT MARGIN?
                  12   12   0AA7  2046          BNEQ    66$                      ; BR IF NORMAL BACKSPACE NEEDED
                        0AA9  2047 ;
```

```
                        0AA9  2048 ; FORCE CONTROL R FOR RUBOUT RESPONSE
                        0AA9  2049 ;
                        0AA9  2050 60$:
      54   78 A5   D0  0AA9  2051         MOVL    UCB$L_SVAPTE(R5),R4      ; GET READ PACKET ADDRESS
           3C A4   B5  0AAD  2052         TSTW    TTY$W_RB_TXTOFF(R4)      ; DATA PRESENT?
              03   12  0AB0  2053         BNEQ    63$                      ; IF DATA PRESENT, ECHO IT IN NORMAL FASSION
            FA71   31  0AB2  2054 62$:     BRW     MOVEREADATA              ; IF NO DATA THEN NO RESPONSE
            FF93   31  0AB5  2055 63$:     BRW     120$                     ; DATA TO ECHO THEN ECHO
            0290   31  0AB8  2056 65$:     BRW     EDITREAD                 ; START THE OUTPUT
                        0ABB  2057 ;
                        0ABB  2058 ; NORMAL BACKSPACE RUBOUT RESPONSE
                        0ABB  2059 ;
      53   20   9A  0ABB  2060 66$:     MOVZBL  #TTY$C_BLANK,R3          ; ASSUME SPACE IS FIRST CHARACTER
42 A5 00FC C5   B1  0ABE  2061         CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5); AND EXTREME RIGHT?
              74   1E  0AC4  2062         BGEQU   100$                     ; IF YES THEN ONLY OUTPUT SPACE
                        0AC6  2063 ;
                        0AC6  2064 ; DELETE IS IN MID SCREEN
                        0AC6  2065 ;
   54   0000'CF   9E  0AC6  2066         MOVAB   W^TTY$A_SPACEBACK,R4     ; ADDRESS BACKSPACE STRING
 00DC C5    84   9B  0ACB  2067         MOVZBW  (R4)+,UCB$W_TT_MULTILEN(R5);AND THEN IT'S LENGTH
      53    08   9A  0AD0  2068 70$:     MOVZBL  #TTY$C_BS,R3             ; START STRING WITH BACKSPACE
            15   11  0AD3  2069         BRB     90$                      ; START OUTPUT
5D 04 A2    01   E2  0AD5  2070 75$:     BBSS    #TTY$V_ST_DEL,4(R2),95$  ; IF NOT FIRST TIME THEN OUTPUT CHARACTER
   54    18 A5   9E  0ADA  2071         MOVAB   UCB$W_BUFQUO(R5),R4      ; ADDRESS STRING SPACE
 00DC C5    01   B0  0ADE  2072         MOVW    #1,UCB$W_TT_MULTILEN(R5)         ; SETUP THE LENGTH OF THE MULTIECHO
      64    53   9B  0AE3  2073         MOVZBW  R3,(R4)                  ; AND PUT THE CHARACTERS THERE
                        0AE6  2074 ;
                        0AE6  2075 ; OUTPUT "\" TO START OR END DELETE SEQUENCE
                        0AE6  2076 ;
      53   5C 8F   9A  0AE6  2077 80$:     MOVZBL  #^A@\@,R3                ; FOR FIRST TIME OUTPUT "\"
 00D8 C5    54   D0  0AEA  2078 90$:     MOVL    R4,UCB$L_TT_MULTI(R5)    ; ADDRESS STRING
                        0AEF  2079         SET_STATE MULTI                  ;
   54   78 A5   D0  0AF3  2080         MOVL    UCB$L_SVAPTE(R5),R4      ; RESTORE THE READ PACKET ADDRESS
        32 A4   B5  0AF7  2081         TSTW    TTY$W_RB_LINREST(R4)     ; ANY CHARACTERS TO THE RIGHT
           3B   13  0AFA  2082         BEQL    95$                      ; NO THEN HANDLE NORMALY
                        0AFC  2083
           3F   BB  0AFC  2084         PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; SAVE NECESSARY STATE
                        0AFE  2085         IF_STATE NOECHO,92$
                        0B02  2086         SET_STATE <EDITREAD>             ; SET THE EDIT ECHOING STATE
   51   30 A4   3C  0B06  2087 92$:     MOVZWL  TTY$W_RB_LINOFF(R4),R1   ; GET THE OFFSET TO THIS CHARACTER
14 A4   2C B441   9E  0B0A  2088         MOVAB   @TTY$L_RB_LIN(R4)[R1],TTY$L_RB_ECHSTR(R4); SETUP THE ADDRESS TO STAR
      53   14 B4   9A  0B10  2089         MOVZBL  @TTY$L_RB_ECHSTR(R4),R3 ; GET THE CHARACTER BACK
      44 A4   01   9B  0B14  2090         MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); CLEAR TO THE END OF LINE THEN E
   0B A4   32 A4   90  0B18  2091         MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH
           51   01   CE  0B1D  2092         MNEGL   #1,R1                    ; setup to find backward size
              042F   30  0B20  2093         BSBW    CHAR_SIZE                ; FIND THE SIZE OF THIS CHARACTER
38 A4 00FC C5   53   A3  0B23  2094         SUBW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4)       ; SUBTRACT OUT THE L
           52   01   9A  0B2A  2095         MOVZBL  #1,R2                    ; SETUP R2 AS 1 CHARACTER
        32 A4   28  0B2D  2096         MOVC3   TTY$W_RB_LINREST(R4),-
14 B4   14 B442         0B30  2097                 @TTY$L_RB_ECHSTR(R4)[R2],@TTY$L_RB_ECHSTR(R4); COMPRESS THE STRING
           3F   BA  0B35  2098         POPR    #^M<R0,R1,R2,R3,R4,R5>   ; SAVE NECESSARY STATE
                        0B37  2099
            F64D   31  0B37  2100 95$:     BRW     FORMAT_CHAR              ; START UP OUTPUT
                        0B3A  2101 100$:
         00FC C5   B7  0B3A  2102         DECW    UCB$W_TT_CURSOR(R5)      ; ADJUST CURSOR
            F607   31  0B3E  2103         BRW     OUTPUTANDWAIT            ; VECTOR FOR SPACE ONLY RESPONSE
```

```
                     0B41  2105
                     0B41  2106 ;++
                     0B41  2107 ; DELETE AN ESCAPE SEQUENCE IN PROGRESS
                     0B41  2108 ;
                     0B41  2109 ; THIS ROUTINE IS USED BY THE CONTROL U AND DELETE LOGIC TO RESET
                     0B41  2110 ; AND DELETE THE ESCAPE SEQUENCE IN PROGRESS
                     0B41  2111 ;
                     0B41  2112 DELESCAPE:                                      ; DELETE CURRENT ESCAPE SEQUENCE
   51    58 A5   D0  0B41  2113          MOVL    UCB$L_IRP(R5),R1              ; GET ADDRESS OF PACKET.
 3C A4   3A A1   A2  0B45  2114          SUBW    IRP$L_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); ADJUST TRANSFER SIZE FOR SEQU
         38 A1   D4  0B4A  2115          CLRL    IRP$L_MEDIA(R1)               ; ESCAPE LENGTH
                     0B4D  2116          CLR_STATE <ESC,BADESC>                ; CLEAR ESCAPE
                 05  0B53  2117 10$:     RSB                                   ; RETURN
```

```
                        0B54  2119 ;++
                        0B54  2120 ; DELETE_WORD
                        0B54  2121 ;
                        0B54  2122 ; DESCRIPTION:
                        0B54  2123 ;       DELETE WORD TO THE LEFT OF THE CURSOR
                        0B54  2124 ;--
                FE6C  31 0B54  2125 GODEL:  BRW     DELCHAR                 ; ACT AS IF A DELETE WAS TYPED
                F73F  31 0B57  2126 DROP_CHAR:BRW   DROP                    ; DROP THE CHARACTER
                        0B5A  2127 DELETE_WORD:
          53  30 A4  3C 0B5A  2128         MOVZWL  TTY$W_RB_LINOFF(R4),R3  ; GET THE OFFSET
                53  D7 0B5E  2129         DECL    R3                      ; SUBTRACT 1
                F5  19 0B60  2130         BLSS    DROP_CHAR               ; no characters then handle suchly
       53  2C B443  9A 0B62  2131         MOVZBL  @TTY$L_RB_LIN(R4)[R3],R3; GET THE CHARACTER AT THIS POINT
E5 00000000'EF  53  E0 0B67  2132         BBS     R3,TTY$A_PREFIX,GODEL; IF THIS CHARACTER
                        0B6F  2133                                         ; IS A PREFIX THEN DELETE ONLY IT
                37  BB 0B6F  2134         PUSHR   #^M<R0,R1,R2,R4,R5>
          50  30 A4  3C 0B71  2135         MOVZWL  TTY$W_RB_LINOFF(R4),R0  ; GET THE CURRENT POSITION
                50  DD 0B75  2136         PUSHL   R0                      ; SAVE THE OFFSET
             51  01  CE 0B77  2137         MNEGL   #1,R1                   ; WE ARE MOVING BACKWARDS
                52  D4 0B7A  2138         CLRL    R2                      ; CLEAN OUT THE CHARACTER POSITION COUNTER
                50  D7 0B7C  2139 10$:     DECL    R0                      ; MOVE ONE CHARACTER TO THE LEFT
                34  19 0B7E  2140         BLSS    30$                     ; WHEN DONE SAY SO
       53  2C B440  9A 0B80  2141         MOVZBL  @TTY$L_RB_LIN(R4)[R0],R3; GET A CHARACTER
0C 00000000'EF  53  E1 0B85  2142         BBC     R3,TTY$A_WORDTERM,20$   ; SKIP OVER TERMINATORS
          30 A4  50  B0 0B8D  2143         MOVW    R0,TTY$W_RB_LINOFF(R4)  ; UPDATE LINOFF TO MAKE TAB DELETION WORK
             03BE  30 0B91  2144         BSBW    CHAR_SIZE               ; GET THE LENGTH OF THIS CHARACTER
             52  53  C0 0B94  2145         ADDL    R3,R2                   ; UPDATE CURSOR POSITION
                E3  11 0B97  2146         BRB     10$                     ; AND CONTINUE
                        0B99  2147
          30 A4  50  B0 0B99  2148 20$:     MOVW    R0,TTY$W_RB_LINOFF(R4)  ; UPDATE LINOFF TO MAKE TAB DELETION WORK
             03B2  30 0B9D  2149         BSBW    CHAR_SIZE               ; GET THE LENGTH OF THIS CHARACTER
             52  53  C0 0BA0  2150         ADDL    R3,R2                   ; UPDATE THE POSITION
                50  D7 0BA3  2151         DECL    R0                      ; KEEP MOVING
                0D  19 0BA5  2152         BLSS    30$                     ; UNTILL THE END
       53  2C B440  9A 0BA7  2153         MOVZBL  @TTY$L_RB_LIN(R4)[R0],R3; GET ANOTHER CHARCTER
E5 00000000'EF  53  E1 0BAC  2154         BBC     R3,TTY$A_WORDTERM,20$   ; KEEP MOVING UNTILL NOT A TERMINATOR
                        0BB4  2155
                50  D6 0BB4  2156 30$:     INCL    R0                      ; GO FOWARD A CHARACTER
        0100 C5  52  B0 0BB6  2157         MOVW    R2,UCB$W_TT_BSPLEN(R5)  ; ECHO THE RIGHT NUMBER OF BACKSPACES
38 A4   00FC C5  52  A3 0BBB  2158         SUBW3   R2,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); SETUP THE CURSOR POSITIO
                51 8ED0 0BC2  2159         POPL    R1                      ; GET THE CURRENT POSITION
        52  51  50  A3 0BC5  2160         SUBW3   R0,R1,R2                ; GET THE NUMBER OF CHARACTERS DELETED
          3C A4  52  A2 0BC9  2161         SUBW    R2,TTY$W_RB_TXTOFF(R4)  ; AND REMOVE THESE CHARACTERS FROM THE COUNT
     14 A4   2C B440  9E 0BCD  2162         MOVAB   @TTY$L_RB_LIN(R4)[R0],TTY$L_RB_ECHSTR(R4); GET THE ADDRESS FOR UPDAT
           32 A4  28 0BD3  2163         MOVC3   TTY$W_RB_INREST(R4),-   ; DELETE THE WORD
     14 B4   2C B441  0BD6  2164                 @TTY$L_RB_LIN(R4)[R1],@TTY$L_RB_ECHSTR(R4);
                37  BA 0BDB  2165         POPR    #^M<R0,R1,R2,R4,R5>     ; RESTORE THE STATE
                        0BDD  2166         IF_STATE NOECHO,40$
                        0BE1  2167         SET_STATE <BACKSPACE,EDITREAD>  ; BACKSPACE OVER THE WORD THEN ECHO EVERYTHI
          44 A4  01  9B 0BE6  2168         MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); SETUP THE MODE FOR EDITECHOING
       0B A4  32 A4  90 0BEA  2169         MOVB    TTY$W_RB_INREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH TO ECHO
             F413  31 0BEF  2170 40$:     BRW     GETNEXTCHAR             ; GET THE NEXT CHARACTER
                        0BF2  2171
```

```
                     0BF2  2173 ;++
                     0BF2  2174 ; Escape_Char
                     0BF2  2175 ;
                     0BF2  2176 ;       Possible escape character found, could be an altmode for
                     0BF2  2177 ; lower case terminals or could be a CSI for new eight bit terminals.
                     0BF2  2178 ; All cases must be handled apropriately.
                     0BF2  2179 ;--
                     0BF2  2180 ESCAPE_CHAR:
   53    7D 8F   91  0BF2  2181         CMPB    #TTY$C_LOWESC1,R3       ; ALT MODE?
                 06  13  0BF6  2182         BEQL    47$                     ; YES
   53    7E 8F   91  0BF8  2183         CMPB    #TTY$C_LOWESC2,R3       ; ALT MODE?
                 08  12  0BFC  2184         BNEQ    52$                     ; YES
39 44 A5         07  E0  0BFE  2185 47$:    BBS     #TT$V_LOWER,UCB$L_DEVDEPEND(R5),51$; LOWER CASE TERMINAL CONTINUE
      53    1B   9A  0C03  2186         MOVZBL  #TTY$C_ESCAPE,R3        ; CHANGE CHARACTER TO ESCAPE
                     0C06  2187 52$:
                     0C06  2188 ;
                     0C06  2189 ; PROCESS ESCAPE SEQUENCES AND ESCAPES
                     0C06  2190 ;
                     0C06  2191         IF_NOT_STATE <ESCAPE,EDITING>,55$; NOT AN ESCAPE TERMINAL THEN EXIT
         F88B    30  0C0E  2192         BSBW    ESCINIT                 ; HANDLE THE ESCAPE PREFIXES
 0103 C5   51    90  0C11  2193         MOVB    R1,UCB$B_TT_ESC(R5)
       48 A4     B4  0C16  2194         CLRW    TTY$W_RB_ESCTKN(R4)
   51  58 A5     D0  0C19  2195         MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
   38 A1  53     B0  0C1D  2196         MOVW    R3,IRP$L_MEDIA(R1)      ; SET TERMINATOR CHARACTER
       3A A1     B6  0C21  2197         INCW    IRP$L_MEDIA+2(R1)       ; START TERMINATOR SIZE
   51  3C A4     3C  0C24  2198         MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE PLACE TO STORE THE ESCAPE
 00 B441   1B    90  0C28  2199         MOVB    #TTY$C_ESCAPE,@TTY$L_RB_TXT(R4)[R1]; AND PLACE THE CHARACTER IN THE
       3C A4     B6  0C2D  2200         INCW    TTY$W_RB_TXTOFF(R4)     ; AND INCREMENT THE WORLD
                     0C30  2201         IF_STATE ESC,60$
                     0C34  2202         SET_STATE ESC                   ; BEGIN ESCAPE SEQUENCE
         F957    31  0C39  2203         BRW     CHECK_BUFFER            ; INSERT THE ESCAPE AT THE END OF THE BUFFER
                     0C3C  2204
                     0C3C  2205 51$:
                     0C3C  2206         IF_NOT_STATE ESC,55$
         FA1C    31  0C40  2207         BRW     ESCINPROG               ; ESCAPE IN PROGGRSS THEN CONTINUE
                     0C43  2208 55$:
         F92B    31  0C43  2209         BRW     INSERT_CHAR             ; ELSE INSERT THE CHARACTER
                     0C46  2210 ;
                     0C46  2211 ; ALREADY IN ESCAPE MODE THEN MAKE THIS BAD
                     0C46  2212 ;
                     0C46  2213 60$:    SET_STATE <BADESC,EOL>          ;
         F3B4    31  0C4E  2214         BRW     GETNEXTCHAR
```

```
                        0C51  2216
                        0C51  2217 ;++
                        0C51  2218 ; FORWARD_CHAR
                        0C51  2219 ;       MOVE THE CURSOR FORWARD 1 CHARACTER POSITION.
                        0C51  2220 ;--
                        0C51  2221 FORWARD_CHAR:
          51   30 A4   3C  0C51  2222          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET TO THE CHARACTER
     53   2C B441   9A  0C55  2223          MOVZBL  @TTY$C_RB_LIN(R4)[R1],R3; GET THE CHARCTER
          32 A4   B7  0C5A  2224          DECW    TTY$W_RB_CINREST(R4)    ; FORWARD A CHARACTER
            27    19  0C5D  2225          BLSS    50$                     ; AT THE END THEN DON'T ALLOW IT
            51    D4  0C5F  2226          CLRL    R1                      ; GOING FORWARD
          02EE    30  0C61  2227          BSBW    CHAR_SIZE               ; CHECK THE SIZE OF THE CHARACTER
38 A4  00FC C5    53  A1  0C64  2228          ADDW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURSOR POSITI
          51   30 A4   3C  0C6B  2229          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET TO THE CHARACTER
     53   2C B441   9A  0C6F  2230          MOVZBL  @TTY$C_RB_LIN(R4)[R1],R3; GET THE CHARCTER
          30 A4   B6  0C74  2231          INCW    TTY$W_RB_CINOFF(R4)     ; UPDATE THE REST POINTER
          53   09   91  0C77  2232          CMPB    #TTY$C_TAB,R3           ; ARE WE FORWARDING OVER A TAB?
            03    12  0C7A  2233          BNEQ    10$                     ; NO THEN CONTINUE
          040E    30  0C7C  2234          BSBW    TABRIGHT                ; YES THEN CHECK FOR A TAB TO THE RIGHT
                        0C7F  2235 10$:    IF_STATE NOECHO,GODROP
           F501   31  0C83  2236          BRW     FORMAT_CHAR             ; THEN OUPUT THE CHARACTER TO GO FOWARD
                        0C86  2237
          32 A4   B4  0C86  2238 50$:    CLRW    TTY$W_RB_LINREST(R4)    ; RESET THE POINTER
           F60D   31  0C89  2239 GODROP: BRW     DROP                    ; AND DROP THE CHARACTER
```

```
                        0C8C  2241 ;++
                        0C8C  2242 ; MOVE_BOL
                        0C8C  2243 ;
                        0C8C  2244 ; DESCRIPTION:
                        0C8C  2245 ;       GO TO THE BEGINNING OF THE LINE.
                        0C8C  2246 ;--
                        0C8C  2247 MOVE_BOL:
              3A A4   B5  0C8C  2248         TSTW    TTY$W_RB_CPZORG(R4)     ; START AT ZERO?
                 03   13  0C8F  2249         BEQL    10$                     ; YES THEN NO SPECAIL ACTION
               0484   30  0C91  2250         BSBW    ZERO_CPZORG             ; ZERO ORG CURSOR IF WE HAVE WRAPPED
        32 A4  30 A4   A0  0C94  2251 10$:    ADDW    TTY$W_RB_LINOFF(R4),TTY$W_RB_LINREST(R4); MOVE THE CHARACTER COUNTS
              30 A4    B4  0C99  2252         CLRW    TTY$W_RB_LINOFF(R4)     ; AND CLEAR THE NUMBER OF CHARACTERS WE HAVE
               03EE   30  0C9C  2253         BSBW    TABRIGHT                ; CHECK FOR TABS TO THE RIGHT
           64  2C A4   D1  0C9F  2254         CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE
                 0C   12  0CA3  2255         BNEQ    20$                     ; NO  THEN JUST DROP IT
           0B A4  01   8E  0CA5  2256         MNEGB   #1,TTY$B_RB_ECHLEN(R4)  ; CLEAN OUT THE STRING TO ECHO
           44 A4  07   9B  0CA9  2257         MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SETUP THE MODE
                       0CAD  2258         SET_STATE <EDITREAD>
              3A A4    B5  0CB1  2259 20$:    TSTW    TTY$W_RB_CPZORG(R4)     ; START AT ZERO?
                 1D   12  0CB4  2260         BNEQ    50$                     ; YES THEN NO SPECAIL ACTION
    00D8 C5  00000001'EF   9E  0CB6  2261         MOVAB   TTY$A_ANSI_DEOL+1,UCB$L_TT_MULTI(R5); GET THE ADDRESS OF A <CR>
          00DC C5  01   D0  0CBF  2262         MOVL    #1,UCB$W_TT_MULTILEN(R5); AND ONLY OUTPUT THE <CR>
                       0CC4  2263         SET_STATE <MULTI,SKIPLF,SKIPCRLF>; SETUP TO DO A MULTIECHO
              38 A4    B4  0CCE  2264         CLRW    TTY$W_RB_CPZCUR(R4)     ; MOVE TO THE BEGINNING WHEN DONE
                 B6   11  0CD1  2265         BRB     GODROP                  ; THEN DISPATCH
                       0CD3  2266 ;
                       0CD3  2267 ; IF THE ORIGION IS NON-ZERO THEN MOVE THE CURSOR TO THE ORIGIONAL POSITION
                       0CD3  2268 ;
        38 A4  3A A4   B0  0CD3  2269 50$:    MOVW    TTY$W_RB_CPZORG(R4),TTY$W_RB_CPZCUR(R4); SETUP TO MOVE BACK
               03CE   31  0CD8  2270         BRW     UPDATE_CURSOR
```

```
                        0CDB  2272 ;++
                        0CDB  2273 ; MOVE_EOL
                        0CDB  2274 ;
                        0CDB  2275 ; DESCRIPTION:
                        0CDB  2276 ;       MOVE TO THE END OF THE LINE.
                        0CDB  2277 ;--
                        0CDB  2278 MOVE_EOL:
   0B A4   32 A4   90   0CDB  2279         MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); GET THE LENGTH TO ECHO
      51   30 A4   3C   0CE0  2280         MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET OF THIS CHARACTER
14 A4   2C B441   9E   0CE4  2281         MOVAB   @TTY$C_RB_LIN(R4)[R1],TTY$L_RB_ECHSTR(R4); SETUP THE ADDRESS ALSO
   44 A4   02     9B   0CEA  2282         MOVZBW  #TTY$K_ER_ECHLINE,TTY$W_RB_MODE(R4); AND SETUP THE MODE
                        0CEE  2283         CLR_STATE TABRIGHT                      ; NO MORE CHARACTERS TO THE RIGHT SO
                        0CF3  2284                                                 ; NO TABS EITHER
                        0CF3  2285         IF_STATE NOECHO,10$                     ; NO ECHO THEN DON'T ECHO
                        0CF7  2286         SET_STATE EDITREAD                      ; THEN THE STATE
   30 A4   32 A4   A0   0CFB  2287 10$:    ADDW    TTY$W_RB_LINREST(R4),TTY$W_RB_LINOFF(R4); FIX THE COUNTS
           32 A4   B4   0D00  2288         CLRW    TTY$W_RB_LINREST(R4)    ;...
            F2FF   31   0D03  2289         BRW     GETNEXTCHAR             ; GET THE NEXT CHARACTER
```

```
                     0D06  2291 ;++
                     0D06  2292 ; QUOTING
                     0D06  2293 ;
                     0D06  2294 ; DESCRIPTION:
                     0D06  2295 ;       INITIATE QUOTING SEQUENCE
                     0D06  2296 ;--
                     0D06  2297 QUOTING:
                     0D06  2298         SET_STATE QUOTING               ;SET THE STATE QUOTE
          F2F7   31  0D0B  2299         BRW     GETNEXTCHAR             ; AND CONTINUE ON
```

```
                     0D0E  2301 ;++
                     0D0E  2302 ; RECALL
                     0D0E  2303 ;
                     0D0E  2304 ; DESCRIPTION:
                     0D0E  2305 ;       SETUP THE NECESSARY STATE TO RECALL THE LAST COMMAND.
                     0D0E  2306 ;--
                     0D0E  2307 RECALL:
28 20 A4    10   E0  0D0E  2308         BBS     #TRM$V_TM_NORECALL,TTY$L_RB_MOD(R4),20$
 54    00E4 C5   D0  0D13  2309         MOVL    UCB$L_TT_TYPAHD(R5),R4  ;GET THE ADDRESS OF THE TYPEAHEA BUFFER
       14 A4     B5  0D18  2310         TSTW    TTY$W_TA_RCLSIZ(R4)     ; CHECK THE SIZE OF THE BUFFER
          1B     13  0D1B  2311         BEQL    10$                     ; DON'T DO ANYTHING IF NO CHAPACTERS
       0E A4     B4  0D1D  2312         CLRW    TTY$W_TA_RCLOFF(R4)     ;CLEAN OUT THE OFFSET TO THE STRING
                 0D20  2313         SET_STATE <RECALL,EDITREAD>     ;TELL THE DISPATCHER THAT WE ARE RECALLING
    54   78 A5   D0  0D24  2314         MOVL    UCB$L_SVAPTE(R5),R4     ; GET THE PACKET ADDRESS
       3C A4     B4  0D28  2315         CLRW    TTY$W_RB_TXTOFF(R4)     ; NO MORE TEXT
       30 A4     B4  0D2B  2316         CLRW    TTY$W_RB_LINOFF(R4)     ; CLEAR THE LINE
       32 A4     B4  0D2E  2317         CLRW    TTY$W_RB_LINREST(R4)    ; EVERYTHING ELSE
 2C A4    64   D0  0D31  2318         MOVL    TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4)
          FB90   31  0D35  2319         BRW     CTRLR                   ; THEN CONTROL-R THE READ
          F2CA   31  0D38  2320 10$:    BRW     GETNEXTCHAR
          F833   31  0D3B  2321 20$:    BRW     INSERT_CHAR
```

```
                        0D3E  2323 ;++
                        0D3E  2324 ; TOGGELINSOV
                        0D3E  2325 ;
                        0D3E  2326 ; Description
                        0D3E  2327 ;       Toggel the insert/overstrike mode flag
                        0D3E  2328 ;--
                        0D3E  2329 TOGGELINSOV:
05 04 A2    17    E4    0D3E  2330          BBSC    #TTY$V_ST_OVERSTRIKE,4(R2),10$; TOGGEL THE OVERSTRIKE BIT
                        0D43  2331          SET_STATE OVERSTRIKE
              F2BA    31    0D48  2332 10$:   BRW     GETNEXTCHAR
```

```
                        0D4B  2335          .SBTTL  EDITREAD - READ EDITING STATE
                        0D4B  2336
                        0D4B  2337 ;++
                        0D4B  2338 ; EDITREAD - STATE TO ALLOW A READ EDITING SEQUENCE
                        0D4B  2339 ;
                        0D4B  2340 ; FUNCTIONAL DESCRIPTION:
                        0D4B  2341 ;
                        0D4B  2342 ;       WHEN ECHOING PARTS OF READS IT IS NECESSARY TO ECHO SEVERAL
                        0D4B  2343 ; STRINGS, THIS ROUTINE FILLS THAT NEED
                        0D4B  2344 ;
                        0D4B  2345 ; INPUTS:
                        0D4B  2346 ;
                        0D4B  2347 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        0D4B  2348 ;       R5 = UCB ADDRESS
                        0D4B  2349 ;
                        0D4B  2350 ; OUTPUTS:
                        0D4B  2351 ;
                        0D4B  2352 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        0D4B  2353 ;       R4 = ADDRESS OF THE READ PACKET
                        0D4B  2354 ;       R5 = UCB ADDRESS
                        0D4B  2355 ;
                        0D4B  2356 ;--
                        0D4B  2357 EDITREAD:                                    ; READ EDITING
          54   78 A5   D0  0D4B  2358          MOVL    UCB$L_SVAPTE(R5),R4     ; GET THE READ PACKET ADDRESS
                        0D4F  2359          CASE    TTY$W_RB_MODE(R4),TYPE=W,<-
                        0D4F  2360                  NORMAL,-
                        0D4F  2361                  CLRECHO,-                   ; ECHO WITH CLEAR BIT SET
                        0D4F  2362                  ECHLINE,-                   ; JUST ECHO WHAT IS SPECIFYED
                        0D4F  2363                  UPDCURSOR,-                 ; UPDATE THE CURSOR POSITION
                        0D4F  2364                  EXITING,-                   ; SETUP TO EXIT.
                        0D4F  2365                  MOVECURSOR,-                ; ALLOW CURSOR TO BE UPDATED
                        0D4F  2366                  CLRREST,-                   ; CLEAR THE REST OF THE LINE
                        0D4F  2367                  PRMECHO,-                   ; ECHO PROMPT THEN CONTINUE
                        0D4F  2368                  PRMECHO1,-                  ; CONTINUING STATE FOR PROMPTING
                        0D4F  2369                  AESECHO,-                   ; ECHO AES STRING AND NOSTRING
                        0D4F  2370                  RVECHO,-                    ; echo in the readverify way
                        0D4F  2371                  SIMCEOL>                    ; SIMULATE CLEAR TO END OF LINE
                        0D6C  2372 NORMAL:
                        0D6C  2373
                        0D6C  2374          IF_STATE <CTRLR,RDVERIFY>,120$          ; HAVE WE COMPLETED THE CONTROL-R
            3A A4    B5 0D78  2375          TSTW    TTY$W_RB_CPZORG(R4)     ; START AT ZERO?
                10   13 0D7B  2376          BEQL    25$                     ; YES THEN NO SPECAIL ACTION
          00FC C5    B5 0D7D  2377          TSTW    UCB$W_TT_CURSOR(R5)     ; AND WE ARE NOW AT ZERO
                0A   12 0D81  2378          BNEQ    25$                     ; YES THEN WE SHOULD FIXUP THE LINE
            3A A4    B4 0D83  2379          CLRW    TTY$W_RB_CPZORG(R4)     ; DO THE FIXING
              0238   30 0D86  2380          BSBW    FIND_BOL                ; FIND THE BEGINNING OF THE LAST LINE
          00FC C5    B4 0D89  2381          CLRW    UCB$W_TT_CURSOR(R5)     ; FIX THE CURSOR (CHANGED DURING FIND_BOL)
                        0D8D  2382 25$:     IF_NOT_STATE SKIPCRLF,119$              ; NO THEN ARE WE DOING A CR <CEL>
                        0D91  2383          SET_STATE <SKIPLF>                      ; YES THEN SKIP A LEADING LINEFEED AND CONTI
          44 A4    06 9B 0D95  2384          MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); THEN CLEAR THE REST
            32 A4    B5 0D99  2385          TSTW    TTY$W_RB_CINREST(R4)    ; CURSOR VALID?
                04   12 0D9C  2386          BNEQ    119$                    ; YES THEN USE IT
          38 A4    01 AE 0D9E  2387          MNEGW   #1,TTY$W_RB_CPZCUR(R4)  ; ELSE USE THE END POSITION
                        0DA2  2388 119$:    SET_STATE <CTRLR,EDITREAD>              ; INDICATE THAT WE ARE OUTPUTTING THE
    34 A4    3C A4    A1 0DAA  2389          ADDW3   TTY$W_RB_TXTOFF(R4),TTY$W_RB_PRMLEN(R4),-
                51        0DAF  2390                  R1                      ; GET THE NUMBER OF CHARACTERS TO OUTPUT
                07   13 0DB0  2391          BEQL    120$                    ; IF NONE THEN DON'T BOTHER.
```

```
                54   4A A4   9E   0DB2  2392          MOVAB   TTY$A_RB_PRM(R4),R4      ; GET THE ADDRESS TO START AT
                      02B5   31   0DB6  2393          BRW     STRTMULTI_1             ; AND START THE STRING ECHOING.
                             0DB9  2394 ;
                             0DB9  2395 ; END OF CONTROL R
                             0DB9  2396 ;
                51   14 A4   D0   0DB9  2397 120$:    MOVL    TTY$L_RB_ECHSTR(R4),R1  ; CHECK FOR A CALLBACK
                        1D   12   0DBD  2398          BNEQ    CALLBACK
                             0DBF  2399 MOVIFNEC:
                     32 A4   B5   0DBF  2400          TSTW    TTY$W_RB_LINREST(R4)    ; ANY MORE CHARACTERS ON THE LINE
                        03   13   0DC2  2401          BEQLU   EXITING                 ; NO THEN EXIT
                      0083   31   0DC4  2402          BRW     MOVECURSOR              ; YES THEN MOVE THE CURSOR
                     44 A4   B4   0DC7  2403 EXITING: CLRW    TTY$W_RB_MODE(R4)       ; CLEAN OUT THE EDITING MODE
                     14 A4   D4   0DCA  2404          CLRL    TTY$L_RB_ECHSTR(R4)
                             0DCD  2405          CLR_STATE <EDITREAD,CTRLR,CTRLO,SKIPCRLF,TABEXPAND>
                      F229   31   0DD9  2406          BRW     GETNEXTCHAR
                             0DDC  2407 ;
                             0DDC  2408 ; IT MAY BE NECESSARY TO CALL THE EDITING CODE BACK TO DO SOME WORK OPERATION
                             0DDC  2409 ; THAT REQUIRED A ZERO ORIGIONAL CURSOR POSITION
                             0DDC  2410 ;
                     44 A4   B4   0DDC  2411 CALLBACK:CLRW   TTY$W_RB_MODE(R4)       ; CLEAN OUT THE EDITING MODE
                     14 A4   D4   0DDF  2412          CLRL    TTY$L_RB_ECHSTR(R4)
                             0DE2  2413          CLR_STATE <EDITREAD,CTRLR,CTRLO,SKIPCRLF,TABEXPAND>
                     3A A4   B5   0DEE  2414          TSTW    TTY$W_RB_CPZORG(R4)     ; MAKE SURE WE WON'T LOOP
                        D4   12   0DF1  2415          BNEQ    EXITING                 ; IF NOT SUCCESSFULL IN CLEARING
                             0DF3  2416                                              ; THE ORIGION THEN DROP THE CHARACTER
                        61   17   0DF3  2417          JMP     (R1)
```

```
                     0DF5  2419 ;
                     0DF5  2420 ; ALTERNATE ECHO STRING.
                     0DF5  2421 ;
44 A4    04   B0    0DF5  2422 AESECHO: MOVW   #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); SETUP TO EXIT
51   28 A4   3C    0DF9  2423          MOVZWL  TTY$W_RB_AESLEN(R4),R   ; THE LENGTH OF THE STRING TO ECHO
54   24 A4   D0    0DFD  2424          MOVL    TTY$L_RB_AES(R4),R4     ; AND THE ADDRESS
      026A   31    0E01  2425          BRW     STRTMULTI_1             ; NOW MULTIECHO
```

```
                        0E04  2427 ;
                        0E04  2428 ; Echo and clear to end of the line
                        0E04  2429 ;
                        0E04  2430 CLRECHO:
                        0E04  2431          SET_STATE <TABEXPAND>           ; SET TO EXPAND TABS
44 A4    06     9B      0E08  2432          MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); THEN MOVE THE CURSOR
51   0B A4     9A      0E0C  2433 COMECHO:MOVZBL  TTY$B_RB_ECHLEN(R4),R1  ; SETUP THE NUNMBER OF CHARACTERS
54   14 A4     D0      0E10  2434          MOVL    TTY$L_RB_ECHSTR(R4),R4  ; SETUP FOR MULTIECHO
      0257     31      0E14  2435          BRW     STRTMOLTI_1             ; AND START THE MULTIECHOING
```

```
                        0E17  2437 ;
                        0E17  2438 ; Clear to the end of the line
                        0E17  2439 ;
                        0E17  2440 CLRREST:
      44 A4    05   9B  0E17  2441          MOVZBW  #TTY$K_ER_MOVECURSOR,TTY$W_RB_MODE(R4);
         38 A4    B5  0E1B  2442          TSTW    TTY$W_RB_CPZCUR(R4)     ; NEGITIVE CUR MEANS DON'T UPDATE
               06 18  0E1E  2443          BGEQ    20$                     ;
 38 A4   00FC C5  B0  0E20  2444          MOVW    UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURRENT
 42 A5   00FC C5  B1  0E26  2445 20$:     CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5); ANYTHING TO CLEAR
               1C 13  0E2C  2446          BEQL    MOVECURSOR              ; NOTHING TO CLEAR THEN JUST MOVE THE
                      0E2E  2447                                          ; CURSOR
   0A 48 A5    18 E1  0E2E  2448          BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),30$; IS IT AN ANSI CRT
54    00000000'EF 9E  0E33  2449          MOVAB   TTY$A_ANSICEL,R4        ; GET A CLEAR TO END OF LINE SEQUENCE
            0223 31  0E3A  2450          BRW     STRTMULTI               ; AND MULTIECHO IT OUT
                      0E3D  2451 ; for non-ansi terminals
      44 A4    0B   9B  0E3D  2452 30$:     MOVZBW  #TTY$K_ER_SIMCEOL,TTY$W_RB_MODE(R4); TELL IT TO COME BACK AND SIMULA
            0075 31  0E41  2453          BRW     SIMCEOL
```

```
                        0E44  2455 ;
                        0E44  2456 ; echo the given data then exit
                        0E44  2457 ;
          44 A4   04    9B  0E44  2458 ECHLINE:MOVZBW  #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); EXIT AFTER ECHOING
                  C2    11  0E48  2459          BRB     COMECHO                    ; AND GO TO THE COMMON CODE
```

```
                        0E4A  2461 ;
                        0E4A  2462 ; cause the cursor to be moved to it's origional position
                        0E4A  2463 ;
                        0E4A  2464 MOVECURSOR:
 44 A4    04     9B     0E4A  2465          MOVZBW  #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); SETUP TO EXIT
       0258     31     0E4E  2466          BRW     UPDATE_CURSOR           ; AND UPDATE THE CURSOR
```

```
                     0E51  2468 ;
                     0E51  2469 ; Echo a prompt and a given amount of data then move the cursor to
                     0E51  2470 ; the specifyed position.  This handles the no prompt case correctly.
                     0E51  2471 ;
                     0E51  2472 PRMECHO:
          51   34 A4  B0  0E51  2473          MOVW    TTY$W_RB_PRMLEN(R4),R1  ; GET THE LENGTH OF THE PROMPT
                 11  13  0E55  2474          BEQL    40$                     ; NO PROMPT THEN JUST ECHO SPECIFYED STRING
       44 A4     08  9B  0E57  2475          MOVZBW  #TTY$K_ER_PRMECHO1,TTY$W_RB_MODE(R4); SET THE NEXT STATE
          54   4A A4  9E  0E5B  2476          MOVAB   TTY$A_RB_PRM(R4),R4     ; GET THE PROMPT'S ADDRESS
                     0E5F  2477          SET_STATE <SKIPCRLF,SKIPLF>     ; SKIP THE FREE LINEFEEDS
                0206  31  0E65  2478          BRW     STRTMULTI_1             ; AND MULTIECHO THE PROMPT OUT
                     0E68  2479 ;
                     0E68  2480 ; SETUP THE CORRECT CURSOR POSITION FOR THE NO PROMPT CASE
                     0E68  2481 ;
00FC C5   3A A4  B0  0E68  2482 40$:    MOVW    TTY$W_RB_CPZORG(R4),UCB$W_TT_CURSOR(R5)
```

```
                        0E6E  2484 ;
                        0E6E  2485 ; second part of prompt echoing if necessary.
                        0E6E  2486 ;
                        0E6E  2487 PRMECHO1:
          38 A4   B5    0E6E  2488          TSTW    TTY$W_RB_CPZCUR(R4)      ; SHOULD WE UPDATE THIS CURSOR
             0B   14    0E71  2489          BGTR    50$                      ; NO IF NOT ZERO
          38 A4   B6    0E73  2490          INCW    TTY$W_RB_CPZCUR(R4)      ; WAS THIS -1?
             06   19    0E76  2491          BLSS    50$                      ; NO THEN DON'T RESET THE CURSOR
38 A4  00FC C5   B0    0E78  2492          MOVW    UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); RESET CURSOR POSITION
                        0E7E  2493 50$:     CLR_STATE <SKIPCRLF,SKIPLF>      ; CLEAN THE SKIP STATE UP
          0B A4   95    0E84  2494          TSTB    TTY$B_RB_ECHLEN(R4)      ; DO WE HAVE ANYTHING TO ECHO
             0B   14    0E87  2495          BGTR    70$                      ; YES THEN ECHO THE SPECIFYED STRING
             03   19    0E89  2496          BLSS    60$                      ; LESS THAN 0 EXIT
           FF89   31    0E8B  2497          BRW     CLRREST                  ; ELSE THEN CLEAR THE REST OF THE LINE
          0B A4   94    0E8E  2498 60$:     CLRB    TTY$B_RB_ECHLEN(R4)      ; CLEAN OUT THEE OFFSET JUST IN CASE
           FF33   31    0E91  2499          BRW     EXITING
           FF6D   31    0E94  2500 70$:     BRW     CLRECHO
```

```
                     0E97  2502 ;
                     0E97  2503 ; read verify echoing (echo's a string replacing fill characters with clear characte
                     0E97  2504 ;
     0B A4    97     0E97  2505 RVECHO: DECB    TTY$B_RB_ECHLEN(R4)      ; TAKE THIS CHARACTER OUT
        1A    19     0E9A  2506         BLSS    20$                      ; NO MORE THEN MOVE THE CURSOR
53   14 B4    9A     0E9C  2507         MOVZBL  @TTY$L_RB_ECHSTR(R4),R3  ; GET A CHARACTER
64   14 A4    D1     0EA0  2508         CMPL    TTY$L_RB_ECHSTR(R4),TTY$L_RB_TXT(R4); IS THIS PAST THE PROMPT
        0A    19     0EA4  2509         BLSS    10$                      ; NO THEN DON'T CHANGE CHARACTERS
53   47 A4    91     0EA6  2510         CMPB    TTY$B_RB_RVFFIL(R4),R3   ; IS THIS A FILL CHARACTER
        04    12     0EAA  2511         BNEQ    10$                      ; NO THEN  CONTINUE
53   46 A4    9A     0EAC  2512         MOVZBL  TTY$B_RB_RVFCLR(R4),R3   ; OTHERWISE ECHO THE CLEAR CHARACTER
     14 A4    D6     0EB0  2513 10$:     INCL    TTY$L_RB_ECHSTR(R4)      ; UPDATE THE COUNT
      F2D1    31     0EB3  2514         BRW     FORMAT_CHAR              ; ELSE FORMAT THE CHARACTER
      FF06    31     0EB6  2515 20$:     BRW     MOVIFNEC                 ; MOVE IF NECESSARY
                     0EB9  2516
```

```
                          0EB9  2518 ;
                          0EB9  2519 ; NON-ANSI CRT THEN SIMULATE CLEAR TO END OF LINE
                          0EB9  2520 ;
                          0EB9  2521 SIMCEOL:
                    0C    E1    0EB9  2522          BBC     #TT$V_SCOPE,-               ; CLEAR TO END OF LINE IS MEANINGLESS
                8C 44 A5          0EBB  2523                  UCB$L_DEVDEPEND(R5),MOVECURSOR; ON HARD COPY TERMINALS
53   42 A5   00FC C5    A3    0EBE  2524          SUBW3   UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3
                08    53    B1    0EC5  2525          CMPW    R3,#8                       ; OUTPUT IN GROUPS OF 8
                    0A    15    0EC8  2526          BLEQ    10$                         ;...
    54   00000000'EF    9E    0ECA  2527          MOVAB   TTY$A_TAB,R4                ; SETUP TO ECHO 8
                018C    31    0ED1  2528          BRW     STRTMOLTI
                          0ED4  2529
         44 A4    05    9B    0ED4  2530 10$:     MOVZBW  #TTY$K_ER_MOVECURSOR,TTY$W_RB_MODE(R4); NEXT MOVE THE CURSOR
       51   53    01    C3    0ED8  2531          SUBL3   #1,R3,R1                    ; SUBTRACT 1 FROM THE COUNT
    54   00000001'EF    9E    0EDC  2532          MOVAB   TTY$A_TAB+1,R4              ;.
                0188    31    0EE3  2533          BRW     STRTMOLTI_1                 ; AND ECHO IT
```

```
                                                J 10
                     0EE6  2535 ;
                     0EE6  2536 ; set the current cursor position
                     0EE6  2537 ;
                     0EE6  2538 UPDCURSOR:
00FC C5   38 A4  B0  0EE6  2539         MOVW    TTY$W_RB_CPZCUR(R4),UCB$W_TT_CURSOR(R5); UPDATE THE CURSOR POSITION
            FED8  31  0EEC  2540         BRW     EXITING                         ; THEN EXIT
```

```
                     0EEF  2542          .SBTTL  Read service routines
                     0EEF  2543 ;++
                     0EEF  2544 ; BACK_TAB
                     0EEF  2545 ;
                     0EEF  2546 ;       Calculates the cursor postion of the beginning of a tab while
                     0EEF  2547 ; at the end of the tab.
                     0EEF  2548 ;
                     0EEF  2549 ; Implicit inputs:
                     0EEF  2550 ;
                     0EEF  2551 ;       R5      UCB address
                     0EEF  2552 ;       TTY$W_RB_LINOFF the current location off of LIN
                     0EEF  2553 ;       TTY$L_RB_LIN the address of the beginning of this line
                     0EEF  2554 ;       TTY$L_RB_TXT the address of the beginning of the data buffer
                     0EEF  2555 ;       TTY$A_RB_PRM offset from svapte to the beginning of the prompt
                     0EEF  2556 ;       TTY$W_RB_CPZORG cursor postion when the read started
                     0EEF  2557 ;
                     0EEF  2558 ; Outputs:
                     0EEF  2559 ;       R3      Mod 8 of the cursor position
                     0EEF  2560 ;--
                     0EEF  2561 BACK_TAB:
           54    DD  0EEF  2562          PUSHL   R4                      ; SAVE R4
     54    78 A5   D0  0EF1  2563          MOVL    UCB$L_SVAPTE(R5),R4     ; GET THE ADDRESS OF THE READ BUFFER
     53    30 A4   3C  0EF5  2564          MOVZWL  TTY$W_RB_LINOFF(R4),R3  ; GET THE CURRENT LOCATION
  53    2C B443    9E  0EF9  2565          MOVAB   @TTY$L_RB_LIN(R4)[R3],R3; POINT TO END OF DATA
           7E    D4  0EFE  2566          CLRL    -(SP)                   ; SET UP CURSOR COUNTER
     64    2C A4   D1  0F00  2567          CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE?
           06    13  0F04  2568          BEQL    10$                     ; YES THEN CONTINUE
     54    2C A4   D0  0F06  2569          MOVL    TTY$L_RB_LIN(R4),R4     ; ELSE JUST USE THE BEGINNING OF THIS ONE
           04    11  0F0A  2570          BRB     20$                     ; AND CONTINUE ON
     54    4A A4   9E  0F0C  2571 10$:     MOVAB   TTY$L_RB_DATA(R4),R4    ; POINT TO DATA START
        54    53   D1  0F10  2572 20$:     CMPL    R3,R4                   ; BUFFER START?
              23   13  0F13  2573          BEQL    40$                     ; THEN REFERENCE POINT FOUND
        0D    73   91  0F15  2574          CMPB    -(R3),#TTY$C_CR         ; CARRIAGE RETURN?
              2C   13  0F18  2575          BEQL    50$                     ; IF EQL THEN REFERENCE POINT FOUND
        09    63   91  0F1A  2576          CMPB    (R3),#TTY$C_TAB         ; TAB?
              27   13  0F1D  2577          BEQL    50$                     ; IF EQL THEN REFERENCE POINT FOUND
        1B    63   91  0F1F  2578          CMPB    (R3),#TTY$C_ESCAPE      ; IS THIS AN ESCAPE
              0F   13  0F22  2579          BEQL    30$                     ; YES THEN REMOVE THE ESCAPE SEQUENCE
  9B    8F    63   91  0F24  2580          CMPB    (R3),#TTY$C_CSI         ; CSI'S COUNT ALSO
              09   13  0F28  2581          BEQL    30$
        20    63   91  0F2A  2582          CMPB    (R3),#TTY$C_BLANK       ; CURSOR CHANGE CHARACTER?
              E1   1F  0F2D  2583          BLSSU   20$                     ; IF LSSU THEN NO
              6E   D6  0F2F  2584          INCL    (SP)                    ; ADJUST FAKE CURSOR
              DD   11  0F31  2585          BRB     20$                     ; CONTINUE
                       0F33  2586 ;
                       0F33  2587 ; ESCAPE INTRODUCER FOUND
                       0F33  2588 ;
           005C    30  0F33  2589 30$:     BSBW    ESCAPE_TAB              ; REMOVE THE ESCAPE SEQUENCE
              D8   11  0F36  2590          BRB     20$                     ; THEN CONTINUE
                       0F38  2591 ;
                       0F38  2592 ; RAN OUT OF BUFFER WITHOUT FINDING A REFERENCE POINT
                       0F38  2593 ;
  53    78 A5   D0  0F38  2594 40$:     MOVL    UCB$L_SVAPTE(R5),R3     ; GET THE ADDRESS OF THE READ BUFFER
  2C A3    63   D1  0F3C  2595          CMPL    TTY$L_RB_TXT(R3),TTY$L_RB_LIN(R3); THIS IS NOT THE FIRST LINE
              04   12  0F40  2596          BNEQ    50$                     ; USE ZERO BASED ORIGION.
  6E    3A A3   A0  0F42  2597          ADDW    TTY$W_RB_CPZORG(R3),(SP); ELSE USE ORITIONAL ORIGION
                       0F46  2598 ;
```

```
                        0F46  2599 ; FOUND THE REFERENCE POINT THEN CONTINUE
                        0F46  2600 ;
      53   8E   FFFFFFF8 8F    CB  0F46  2601 50$:    BICL3   #^XOFFFFFFF8,(SP)+,R3   ; GET MOD 8 OF CURSOR
                             54 8ED0  0F4E  2602            POPL    R4                  ; RESTORE R4
                                05  0F51  2603            RSB
                                    0F52  2604
```

```
                     0F52  2606 ;++
                     0F52  2607 ; CHAR_SIZE
                     0F52  2608 ;
                     0F52  2609 ; DESCRIPTION:
                     0F52  2610 ;
                     0F52  2611 ;       GIVEN A CHARACTER IN R3 THIS ROUTINE WILL DETERMINE THE NUMBER
                     0F52  2612 ; OF CURSOR POSITIONS THE CHARACTER WILL TAKE UP.
                     0F52  2613 ;
                     0F52  2614 ; INPUTS:
                     0F52  2615 ;       R1 = 0 FOR FOWARD CHARACTER SIZE -1 FOR BACKWARD CHARACTER SIZE
                     0F52  2616 ;       R3 = CHARACTER
                     0F52  2617 ;       R5 = ADDRESS OF THE UCB
                     0F52  2618 ;
                     0F52  2619 ; OUTPUTS:
                     0F52  2620 ;
                     0F52  2621 ;       R3 = THE LENGTH IN CURSOR POSITIONS OF THIS CHARACTER
                     0F52  2622 ;       R5 = UCB ADDRESS
                     0F52  2623 ;
                     0F52  2624 ;--
                     0F52  2625
                     0F52  2626 CHAR_SIZE:
                     0F52  2627         CASE    W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
                     0F52  2628                         10$,-                   ; Back space
                     0F52  2629                         20$,-                   ; tab
                     0F52  2630                         5$,-                    ; linefeed (no action)
                     0F52  2631                         5$,-                    ; Vtab (no action)
                     0F52  2632                         5$,-                    ; form (no action)
                     0F52  2633                         60$>                    ; <CR>
   F0 8F    93     0F65  2634         BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
0000'CF43          0F68  2635         W^TTY$A_TYPE[R3]                ; TEST FOR SPECIAL
       04    12     0F6C  2636         BNEQ    5$                      ; NON SPACING CHARACTER
 53    01    9A     0F6E  2637         MOVZBL  #1,R3                   ; OTHERWISE ONLY 1 CHARACTER POSITION
             05     0F71  2638         RSB
                    0F72  2639
             53     D4    0F72  2640 5$:     CLRL    R3                      ; CLEAR R3 OUT
             05     0F74  2641         RSB                             ; AND RETURN
                    0F75  2642
             53 01  CE    0F75  2643 10$:    MNEGL   #1,R3                   ; BACKSPACE THEN BACKUP
             05     0F78  2644         RSB
                    0F79  2645
                    0F79  2646 20$:                                    ; TAB
53  00FC C5  03 00 EF    0F79  2647         EXTZV   #0,#3,UCB$W_TT_CURSOR(R5),R3; GET HORIZONTAL POINTER
             51     D5    0F80  2648         TSTL    R1                      ; IS THIS A BACKWARD CALCULATION
             03     18    0F82  2649         BGEQ    23$                     ; NO THEN SUBTRACT
          FF68      30    0F84  2650         BSBW    BACK_TAB                ; CALCULATE THE MOD 8 OF THE CURSOR
          53 08     C3    0F87  2651 23$:    SUBL3   R3,#8,R3                ; NORMALIZE THE RESULT
             05     0F8B  2652         RSB                             ; AND RETURN
                    0F8C  2653
          53 00FC C5    3C    0F8C  2654 60$:    MOVZWL  UCB$W_TT_CURSOR(R5),R3  ; CARRIAGE RETURN SHOULD ZERO CURSOR
             05     0F91  2655         RSB                             ; POSITION
```

```
                        0F92  2657 ;++
                        0F92  2658 ; ESCAPE_TAB
                        0F92  2659 ;
                        0F92  2660 ;       Handle the removal of escape sequencses from a prompt
                        0F92  2661 ; when deleting tabs.
                        0F92  2662 ;
                        0F92  2663 ; Implicit inputs:
                        0F92  2664 ;
                        0F92  2665 ;       R3 the address of the escape character
                        0F92  2666 ;       R5 THE UCB ADDRESS
                        0F92  2667 ;       4(SP) the internal count
                        0F92  2668 ;
                        0F92  2669 ; Implicit outputs:
                        0F92  2670 ;
                        0F92  2671 ;       4(SP) updated internal count
                        0F92  2672 ;
                        0F92  2673 ; All registers are saved
                        0F92  2674 ;--
                        0F92  2675 ESCAPE_TAB:
          51    DD      0F92  2676         PUSHL   R1                      ; SAVE A SCRATCH REGISTER
    51    08 AE   D0      0F94  2677         MOVL    8(SP),R1                ; RESTORE THE INTERNAL COUNT
          18    BB      0F98  2678         PUSHR   #^M<R3,R4>              ; SAVE THE REST OF THE REGISTERS
       54 53    D0      0F9A  2679         MOVL    R3,R4                   ; MOVE THE CURRENT POSITION TO R4
       53 84    9A      0F9D  2680         MOVZBL  (R4)+,R3                ; GET THE CHARACTER THAT INTRODUCES THE SEQUENCE
          51    DD      0FA0  2681         PUSHL   R1                              ; SAVE R1
        F4F7    30      0FA2  2682         BSBW    ESCINIT                         ; INIT THE ESCAPE SEQUENCE RULES
0103 C5 51      90      0FA5  2683         MOVB    R1,UCB$B_TT_ESC(R5)             ; ...
          51 8ED0       0FAA  2684         POPL    R1                              ; RESTORE R1
                        0FAD  2685
          51    D7      0FAD  2686 20$:    DECL    R1                      ; DECREMENT THE COUNT
       53 84    9A      0FAF  2687         MOVZBL  (R4)+,R3                ; GET THE CHARACTER AND MOVE TO THE NEXT
        F4F8    30      0FB2  2688         BSBW    ESCSYNTAX               ; CHECK THE SYNTAX
          F6    14      0FB5  2689         BGTR    20$                     ; CONTINUE IF NECESSARY
          18    BA      0FB7  2690         POPR    #^M<R3,R4>              ; SEQUENCE COMPLETE THEN RESTORE THE REGISTERS
08 AE     51    D0      0FB9  2691         MOVL    R1,8(SP)                ; PUT THE COUNT BACK
          51 8ED0       0FBD  2692         POPL    R1                      ; RESTORE THE LAST REGISTER
                05      0FC0  2693         RSB                             ; AND NOW RETURN
```

```
                        OFC1  2695 ;++
                        OFC1  2696 ; FIND_BOL - FIND THE BEGINNING OF THIS LINE
                        OFC1  2697 ;
                        OFC1  2698 ; Description:
                        OFC1  2699 ;
                        OFC1  2700 ;       Given a string this routine will find the offset to the character
                        OFC1  2701 ; that will end up in the first character position of the bottom line
                        OFC1  2702 ; of the screen
                        OFC1  2703 ;
                        OFC1  2704 ;
                        OFC1  2705 ; IMPLICIT INPUTS:
                        OFC1  2706 ;
                        OFC1  2707 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        OFC1  2708 ;       R4 = ADDRESS OF THE READ BUFFER
                        OFC1  2709 ;       R5 = ADDRESS OF THE UCB
                        OFC1  2710 ;       TTY$L_RB_TXT
                        OFC1  2711 ;       TTY$L_RB_LIN
                        OFC1  2712 ;       TTY$L_RB_PRMLEN
                        OFC1  2713 ;       TTY$L_RB_TXTOFF assumed non-zero
                        OFC1  2714 ;       TTY$A_RB_PRM
                        OFC1  2715 ;       TTY$W_RB_LINOFF
                        OFC1  2716 ;       TTY$W_RB_LINREST assumed zero
                        OFC1  2717 ;
                        OFC1  2718 ; IMPLICIT OUTPUTS:
                        OFC1  2719 ;
                        OFC1  2720 ;       TTY$L_RB_LIN address of the first character in this line of data
                        OFC1  2721 ;       TTY$L_RB_LINOFF offset from LIN to the end of the line
                        OFC1  2722 ;       R3 is destroyed.
                        OFC1  2723 ;--
                        OFC1  2724 FIND_BOL:
          00FC C5   B4  OFC1  2725          CLRW    UCB$W_TT_CURSOR(R5)     ; CLEAN THE CURSOR POSITION
                        OFC5  2726 FIND_BOL_NOCLEAR::
              51   D4   OFC5  2727          CLRL    R1                      ; CLEAN R1
          2C A4    64   D0   OFC7  2728          MOVL    TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4); START WITH THE BEGINNING
             00C0 8F   BB   OFCB  2729          PUSHR   #^M<R6,R7>              ; GET A FEW SCRATCH REGISTERS
              56   D4   OFCF  2730          CLRL    R6                      ; CLEAR R6, THE BEGINNING OFFSET
57    3C A4    34 A4   A1   OFD1  2731          ADDW3   TTY$W_RB_PRMLEN(R4),TTY$W_RB_TXTOFF(R4),R7; GET THE ENDING OFFSET
      00 2A A4    00   E5   OFD7  2732          BBCC    #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),210$; CLEAR THE WRAP STATE
              57   56   B1   OFDC  2733 210$:    CMPW    R6,R7                   ; ARE WE DONE?
              6D   18   OFDF  2734          BGEQ    240$                    ; YES THEN EXIT
       53   4A A446   9A   OFE1  2735          MOVZBL  TTY$A_RB_PRM(R4)[R6],R3 ; GET THE CHARACTER
              56   D6   OFE6  2736          INCL    R6                      ; MOVE TO THE NEXT CHARACTER
00000000'EF43    04   91   OFE8  2737          CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]
              3B   13   OFF0  2738          BEQL    250$                    ; HANDLE ESCAPE SEQUENCES
              53   0D   91   OFF2  2739          CMPB    #TTY$C_CR,R3            ; A CR ZEROS CURSOR
              2B   13   OFF5  2740          BEQL    230$                    ; SO DO SO
                 FF58   30   OFF7  2741          BSBW    CHAR_SIZE               ; OTHER WISE GET THECURSOR POSITION
          00FC C5   53   A0   OFFA  2742          ADDW    R3,UCB$W_TT_CURSOR(R5)  ; ADD IN THE POSITION
    42 A5    00FC C5   B1   OFFF  2743          CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5); IS THIS A WRAP
              D5   19   1005  2744          BLSS    210$                    ; NO THEN GO ON
    00FC C5    42 A5   A2   1007  2745          SUBW    UCB$W_DEVBUFSIZ(R5),UCB$W_TT_CURSOR(R5); SUBTRACT OUT THE
                       100D  2746                                          ; LINE LENGTH
       53   4A A446   9E   100D  2747          MOVAB   TTY$A_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THE WRAP
      00 2A A4    00   E2   1012  2748 220$:    BBSS    #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),225$; SET UP THE WRAP CHARACTERI
              64   53   D1   1017  2749 225$:    CMPL    R3,TTY$L_RB_TXT(R4)     ; ARE WE PAST THE PROMPT?
              C0   19   101A  2750          BLSS    210$                    ; YES THEN CONTINUE
          2C A4    53   D0   101C  2751          MOVL    R3,TTY$L_RB_LIN(R4)     ; AND THIS IS THE LINE
```

```
                BA   11  1020  2752          BRB     210$                        ; THEN CONTINUE
                                1022  2753
53   4A A446    9E       1022  2754 230$:    MOVAB   TTY$A_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THE WRAP
     00FC C5    B4       1027  2755          CLRW    UCB$W_TT_CURSOR(R5)     ; CLEAR CURSOR ON A <CR>
           E5   11       102B  2756          BRB     220$                        ; NOW CONTINUE
                         102D  2757
                         102D  2758 ;
                         102D  2759 ; ESCAPE SEQUENCE INTRODUCER
                         102D  2760 ;
           51   DD       102D  2761 250$:    PUSHL   R1                          ; SAVE R1
              F46A   30   102F  2762          BSBW    ESCINIT                     ; INIT THE SYNTAX RULES
0103 C5       51   90     1032  2763          MOVB    R1,UCB$B_TT_ESC(R5)         ;
              51 8ED0     1037  2764          POPL    R1                          ; RESTORE THE REGISTER
                         103A  2765 260$:
     57    56   B1       103A  2766          CMPW    R6,R7                       ; ARE WE FINISHED
              0F   18     103D  2767          BGEQ    240$                        ; YES THEN EXIT
53   4A A446    90       103F  2768          MOVB    TTY$A_RB_PRM(R4)[R6],R3 ; GET THE NXT CHARACTER
              56   D6     1044  2769          INCL    R6                          ; MOVE TO THE NEXT CHARACTER
                 F464   30   1046  2770          BSBW    ESCSYNTAX               ; AND CHECK THE SYNTAX
                   EF   14   1049  2771          BGTR    260$                    ; MORE THEN CONTINUE
                 FF8E   31   104B  2772          BRW     210$                    ; NO MORE THEN GO BACK
                         104E  2773
53   4A A446    9E       104E  2774 240$:    MOVAB   TTY$A_RB_PRM(R4)[R6],R3; GET THE ADDRESS OF THIS CHARACTER
  53    2C A4    C2       1053  2775          SUBL    TTY$L_RB_LIN(R4),R3     ; GET THE OFFSET TO THE LAST CHARACTER
  30 A4    53    B0       1057  2776          MOVW    R3,TTY$W_RB_LINOFF(R4)  ; AND LEAVE IT there
     00C0 8F    BA       105B  2777          POPR    #^M<R6,R7>                  ; RESTORE THE REGISTERS
                05   105F  2778          RSB
```

```
                     1060  2780 ;++
                     1060  2781 ; STRTMULTI - START A MULTIECHO STRING GIVEN THE ADDRESS OF AN ASCIC STRING
                     1060  2782 ;
                     1060  2783 ; FUNCTIONAL DESCRIPTION:
                     1060  2784 ;
                     1060  2785 ; THIS ROUTINE STARTS THE SPECIFIED MULTIECHO STRING ON A UNIT.
                     1060  2786 ; THE COUNT OF THE STRING IS PLACED IN UCB$W_TT_MULTILEN, AND THE
                     1060  2787 ; ADDRESS OF THE FIRST CHARACTER IN THE STRING IS PLACED IN UCB$L_TT_MULTI.
                     1060  2788 ; THEN STATE MULTI IS SET AND MULTIECHO IS CALLED.
                     1060  2789 ;
                     1060  2790 ; STRTMULTI_1 - starts a multiecho string given the address and a length
                     1060  2791 ;                of a string (address in R4, length in R1)
                     1060  2792 ;
                     1060  2793 ; INPUTS
                     1060  2794 ;
                     1060  2795 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     1060  2796 ;       R4 = ADDRESS OF A COUNTED MULTIECHO STRING (ASCIC)
                     1060  2797 ;       R5 = UCB ADDRESS
                     1060  2798 ;
                     1060  2799 ; OUTPUTS:
                     1060  2800 ;
                     1060  2801 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     1060  2802 ;       R3 = FIRST CHARACTER
                     1060  2803 ;       R5 = UCB ADDRESS
                     1060  2804 ;       R4 DESTROYED
                     1060  2805 ;--
                     1060  2806 STRTMULTI::
00DE C5   00DC C5    B0   1060  2807         MOVW    UCB$W_TT_MULTILEN(R5),UCB$W_TT_SMLTLEN(R5); SAVE LENGTH
     00DC C5   84    9B   1067  2808         MOVZBW  (R4)+,UCB$W_TT_MULTILEN(R5)     ;GET THE LENGTH INTO ALTLENA
               0C    11   106C  2809         BRB     LOADADR
                     106E  2810 STRTMULTI_1::
00DE C5   00DC C5    B0   106E  2811         MOVW    UCB$W_TT_MULTILEN(R5),UCB$W_TT_SMLTLEN(R5); SAVE LENGTH
     00DC C5   51    B0   1075  2812         MOVW    R1,UCB$W_TT_MULTILEN(R5)        ; MOVE IN THE NEW LENGTH
                     107A  2813 LOADADR:
00E0 C5   00D8 C5    D0   107A  2814         MOVL    UCB$L_TT_MULTI(R5),UCB$L_TT_SMLT(R5); SAVE THE ADDRESS
     00D8 C5   54    D0   1081  2815         MOVL    R4,UCB$L_TT_MULTI(R5)           ;THEN THE ADDRESS INTO MULTI
                     1086  2816         SET_STATE <MULTI>                       ;NEXT SET MULTIECHO STATE
               F3D1  31   108A  2817         brw     multiechoing
```

```
                        108D  2819 ;++
                        108D  2820 ; TABRIGHT - CHECK FOR A TAB TO THE RIGHT OF THE CURSOR POSITION
                        108D  2821 ;
                        108D  2822 ; DESCRIPTION:
                        108D  2823 ;
                        108D  2824 ;       This routine will detect a tab character to the right of the cursor
                        108D  2825 ; position and set a flag indicating that insert mode is illegal.  This
                        108D  2826 ; prevents having to recalculate tab and wrap position for each character.
                        108D  2827 ;
                        108D  2828 ; Inputs:
                        108D  2829 ;       R4 - Read packet address
                        108D  2830 ;
                        108D  2831 ; Outputs:
                        108D  2832 ;
                        108D  2833 ;       TABRIGHT state bit set if a tab is to the right of the cursor
                        108D  2834 ;               reset if not.
                        108D  2835 ;       All registers preserved.
                        108D  2836 ;--
                        108D  2837 TABRIGHT:
                03   BB 108D  2838         PUSHR   #^M<R0,R1>              ; SAVE SOME WORKING REGISTERS
                        108F  2839         CLR_STATE TABRIGHT              ; CLEAR THS STATE
          51    30 A4 3C  1094  2840         MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET INTO THE STRING
2C B441   32 A4   09 3A 1098  2841         LOCC    #TTY$C_TAB,TTY$W_RB_LINREST(R4),@TTY$L_RB_LIN(R4)[R1];
                        109F  2842                                         ; TRY TO FIND A TAB
                05   13 109F  2843         BEQL    10$                     ; DID WE FIND A TAB?
                        10A1  2844         SET_STATE TABRIGHT              ; YES THEN SET THE STATE
                03   BA 10A6  2845 10$:    POPR    #^M<R0,R1>              ; RESTORE THE REGISTERS
                   05 10A8  2846         RSB                             ; AND RETURN
                        10A9  2847
```

```
                        10A9  2849 ;++
                        10A9  2850 ; UPDATE_CURSOR
                        10A9  2851 ;
                        10A9  2852 ; DESCRIPTION:
                        10A9  2853 ;
                        10A9  2854 ;       WILL SETUP THE NECESSARY SEQUENCE OF EVENTS TO HAVE
                        10A9  2855 ; THE CURSOR MVOVED TO THE PLACE SPECIFYED BY TTY$W_RB_CPZCUR
                        10A9  2856 ;
                        10A9  2857 ; INPUTS:
                        10A9  2858 ;
                        10A9  2859 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        10A9  2860 ;       R5 = UCB ADDRESS
                        10A9  2861 ;       TTY$W_RB_CPZCUR = THE FINAL CURSOR POSITION
                        10A9  2862 ;       UCB$W_TT_CURSOR = THE CURRENT POSITION.
                        10A9  2863 ;
                        10A9  2864 ; OUTPUT:
                        10A9  2865 ;
                        10A9  2866 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        10A9  2867 ;       R4 = ADDRESS OF THE READ PACKET
                        10A9  2868 ;       R5 = UCB ADDRESS
                        10A9  2869 ; R3 DESTROYED
                        10A9  2870 ;--
                        10A9  2871 UPDATE_CURSOR:
54   78 A5    D0        10A9  2872         MOVL    UCB$L_SVAPTE(R5),R4             ; GET THE READ PACKET ADDRESS
     38 A4    A3        10AD  2873         SUBW3   TTY$W_RB_CPZCUR(R4),-           ; SUBTRACT THE FINAL CURSOR POSITION
00FC C5                 10B0  2874                 UCB$W_TT_CURSOR(R5),-           ; FROM THE CURRENT CURSOR POSITION
0100 C5                 10B3  2875                 UCB$W_TT_BSPLEN(R5)             ; AND ECHO THAT MANY BACKSPACES
00FC C5    B1           10B6  2876         CMPW    UCB$W_TT_CURSOR(R5),-
  42 A5                 10BA  2877                 UCB$W_DEVBUFSIZ(R5)             ; ARE WE AT THE END OF THE LINE
     0A    1F           10BC  2878         BLSSU   5$                              ; NO THEN CONTINUE
00FC C5    B7           10BE  2879         DECW    UCB$W_TT_CURSOR(R5)             ; KEEP THE POSITION CORRECT
0100 C5    B7           10C2  2880         DECW    UCB$W_TT_BSPLEN(R5)             ; ELSE THE TERMINAL DOES A BACKSPACE
     4D    13           10C6  2881         BEQL    40$                             ; NO BACKSPACES THEN EXIT
                       10C8  2882 5$:
   45 48 A5    18       10C8  2883         BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),30$; NOT AN ANSI TERMINAL THEN BR
    0100 C5    08       10CD  2884         CMPW    #8,UCB$W_TT_BSPLEN(R5)          ; CAN WE OPTOMIZE ON AN ANSI TERMINA
     3E    14           10D2  2885         BGTR    30$                             ; NO THEN OUTPUT BACKSPACES
     3F    BB           10D4  2886         PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; SAVE THE REGISTERS
 51   00000000'EF       10D6  2887         MOVZBL  TTY$A_ANSIBACKUP,R1             ; GET THE LENGTH OF THE STRING
     51    D6           10DD  2888         INCL    R1                              ; ADD IN THE COUNT TOO.
0C A4   00000000'EF     51    28  10DF  2889         MOVC3   R1,TTY$A_ANSIBACKUP,TTY$Q_RB_ECHOAREA(R4); MOVE THE CHARACTER
                        10E8  2890                                                 ; STRING IN TO PLACE
     3F    BA           10E8  2891         POPR    #^M<R0,R1,R2,R3,R4,R5>          ; RESTORE THE REGISTERS
     7E    D4           10EA  2892         CLRL    -(SP)                           ; GET ENOUGH ROOM FOR THE DIVIDEND
   7E   0100 C5    3C    10EC  2893         MOVZWL  UCB$W_TT_BSPLEN(R5),-(SP)       ;  AND PUT IT THE DIVISOR
     51   12 A4    9E    10F1  2894         MOVAB   TTY$Q_RB_ECHOAREA+6(R4),R1      ; GET THE ADDRESS OF THE FISRST BYTE
53   6E   6E      0A    7B    10F5  2895 10$:     EDIV    #10,(SP),(SP),R3        ; GET THE FIRST DIGIT
          71     53    80    10FA  2896         ADDB    R3,-(R1)                    ; PUT IT IN PLACE
                 6E    D5    10FD  2897         TSTL    (SP)                        ; DO WE HAVE ANY MORE TO DO
                 F4    12    10FF  2898         BNEQ    10$                         ; YES THEN DO SO
                 8E    D5    1101  2899         TSTL    (SP)+                       ; CLEAN THE STACK
                 8E    D5    1103  2900         TSTL    (SP)+                       ;
 00FC C5   38 A4    B0    1105  2901         MOVW    TTY$W_RB_CPZCUR(R4),UCB$W_TT_CURSOR(R5); UPDATE THE CURSOR POSITION
                        110B  2902 ;       MOVZBW  #TTY$K_ER_UPDCURSOR,TTY$W_RB_MODE(R4); SETUP THE EDITREAD STATE
                        110B  2903 ;       SET_STATE EDITREAD                      ; And setup the edit read state
     54   0C A4    9E    110B  2904         MOVAB   TTY$Q_RB_ECHOAREA(R4),R4        ; GET THE ECHO AREA ADDRESS
            FF4E    31    110F  2905         BRW     STRTMOLTI                       ; and start the multiechoing
```

```
                        1112  2906
                        1112  2907 30$:     SET_STATE <BACKSPACE>                   ; AND START MULTIECHOING BACKSPACES
EEED    31              1115  2908 40$:     BRW       GETNEXTCHAR                   ; GET THE NEXT CHARACTER
```

```
                        1118  2910 ;++
                        1118  2911 ;ZERO_CPZORG - HANDLE CASE WHERE WE NEED A FRESH LINE
                        1118  2912 ;
                        1118  2913 ; DESCRIPTION:
                        1118  2914 ;
                        1118  2915 ;       This routine will check and make sure that the origional cursor
                        1118  2916 ; position becomes zero.  This will then callback the routine that called it
                        1118  2917 ; and re-run the operation.
                        1118  2918 ;
                        1118  2919 ; Inputs:
                        1118  2920 ;
                        1118  2921 ;       R4 - READ BUFFER ADDRESS
                        1118  2922 ;       R5 - UCB ADDRESS
                        1118  2923 ;       (SP) - ADDRESS OF A ROUTINE TO CONTINUE WITH
                        1118  2924 ;--
                        1118  2925 ZERO_CPZORG:
06 2A A4    00    E0    1118  2926         BBS     #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),50$; IF WE WRAPPED THEN
                        111D  2927                                                 ; WE MUST FIX UP THE IO
   2C A4    64    D1    111D  2928         CMPL    TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4); DID WE WRAP?
            07    13    1121  2929         BEQL    60$                            ; NO THEN NO SPECAIL ACTION
   14 A4    8E    D0    1123  2930 50$:     MOVL    (SP)+,TTY$L_RB_ECHSTR(R4); SETUP A CALLBACK
         F79E    31    1127  2931         BRW     CTRLR                          ; AND REFRESH THE IO
                 05    112A  2932 60$:     RSB
```

```
                        112B  2934          .sbttl  Read passall
                        112B  2935 ;++
                        112B  2936 ;
                        112B  2937 ; PASS ALL  AND NO FILTER HANDELER
                        112B  2938 ;
                        112B  2939 ; INPUTS:
                        112B  2940 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        112B  2941 ;       R3 = CHARACTER TO INSERT
                        112B  2942 ;       R4 = READ PACKET ADDRESS
                        112B  2943 ;       R5 = UCB ADDRESS
                        112B  2944 ;
                        112B  2945 ;--
                        112B  2946 PASSALL::
                        112B  2947
      51    3C A4   3C  112B  2948          MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE PLACE TO PUT THIS CHARACTER
   00 B441    53   90  112F  2949          MOVB    R3,@TTY$[_RB_TXT(R4)[R1]; PUT THE CHARACTER AWAY
                      1134  2950          IF_STATE ESC,1800$              ; ESCAPE THE EXIT
                      1138  2951          IF_NOT_STATE PASALL,1900$       ; NOFORMAT THEN UPPER CASING MAY BE NECESSAR
                      113C  2952 1005$:    IF_STATE ESCAPE,1045$           ; IF ESCAPE MODE THEN EXIT
 62 1C B4    53   E0  1140  2953 1040$:    BBS     R3,@TTY$L_RB_TERM(R4),1500$; IS THIS A TERMINATOR
         3C A4   B6  1145  2954          INCW    TTY$W_RB_TXTOFF(R4)     ; INCREMENT THE COUNT
   40 A4 3C A4   B1  1148  2955 1010$:    CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); ARE WE FINSIHED
            46   1E  114D  2956          BGEQU   1600$                   ; YES THEN FINISH
                      114F  2957
                      114F  2958 1030$:    IF_STATE <ESC>,1000$            ; IS THIS THIS ESCAPE
                      1153  2959          IF_NOT_STATE PRE,1025$          ; IF PRETYPEAHEAD THEN RETURN
            05   1157  2960 1020$:    RSB
                      1158  2961 1025$:    IF_NOT_STATE NOECHO,1027$
            00   E1  115C  2962          BBC     #TT2$V_LOCALECHO,-      ; LOCAL ECHO?
      10 48 A5        115E  2963                  UCB$L_DEVDEPND2(R5),1026$; NO THEN MUST BE NOECHO
            06   E0  1161  2964          BBS     #TRM$V_TM_NOECHO,-      ; NOECHO SPECIFYED
      0B 20 A4        1163  2965                  TTY$L_RB_MOD(R4),1026$  ; ON THE READ
          F07E   31  1166  2966          BRW     FORMAT_LOCAL            ; ELSE LOCAL ECHO IT
          F01B   31  1169  2967 1027$:    BRW     FORMAT_CHAR             ; FORMAT THE CHARACTER
                      116C  2968 1000$:    IF_NOT_STATE PRE,1026$
            05   1170  2969          RSB
          EE91   31  1171  2970 1026$:    BRW     GETNEXTCHAR
                      1174  2971
          0065   31  1174  2972 1045$:    BRW     1400$
                      1177  2973 ; CONVERT LOWER TO UPPER IF NOFORMAT
                      1177  2974 ;
      51    58 A5   D0  1177  2975 1900$:    MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
   BC 20 A1    08   E1  117B  2976          BBC     #IO$V_CVTLOW,IRP$W_FUNC(R1),1005$; BR IF CONVERT LOWER TO UPPER
B5 0000'CF43    03   E1  1180  2977          BBC     #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],1005$; BR IF NOT LOWER CHARACTER
         53   20   AA  1187  2978          BICW    #^X020,R3               ; CONVERT TO UPPER CASE
      51    3C A4   3C  118A  2979          MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE PLACE TO PUT THIS CHARACTER
   00 B441    53   90  118E  2980          MOVB    R3,@TTY$[_RB_TXT(R4)[R1]; PUT THE CHARACTER AWAY
            A7   11  1193  2981          BRB     1005$
                      1195  2982
      51    58 A5   D0  1195  2983 1600$:    MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
            38 A1   B4  1199  2984          CLRW    IRP$L_MEDIA(R1)         ; SET NO TERMINATOR
                      119C  2985          SET_STATE EOL                   ; SIGNIFY END OF LINE
   B2 04 A2    1A   E4  11A0  2986          BBSC    #TTY$V_ST_PRE,4(R2),1020$; CLEAR PRE IF NECESSARY AND EXIT
            A8   11  11A5  2987          BRB     1030$
                      11A7  2988
                      11A7  2989
   30 A4    3C A4   B0  11A7  2990 1500$:    MOVW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_LINOFF(R4); SETUP LINOFF
```

```
                                11AC  2991          SET_STATE EOL                   ; SIGNIFY END OF LINE
                     66   11    11B0  2992          BRB     1440$
                                11B2  2993 ;
                                11B2  2994 ; ESCAPE IN PROGRESS
                                11B2  2995 ;
          51    58 A5    D0    11B2  2996 1800$:  MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
                3A A1    B6    11B6  2997         INCW    IRP$L_MEDIA+2(R1)       ; ADD 1 TO THE SIZE
                3C A4    B6    11B9  2998         INCW    TTY$W_RB_TXTOFF(R4)     ; UP THE OFFSET BECAUSE WE PUT 1 CHAR IN THE
                 F2EE    30    11BC  2999         BSBW    ESCSYNTAX               ; CHECK THE SYNTAX OF THIS CHAR
                   18    14    11BF  3000         BGTR    1830$                   ; OK THEN CONTINUE ON
                   04    13    11C1  3001         BEQL    1810$                   ; VALID SEQUENCE CONTINUE
                                11C3  3002
                                11C3  3003         SET_STATE BADESC                ; A BAD ESCAPE SEQUENCE THEN SAY SO
                                11C7  3004 1810$:  CLR_STATE ESC                   ; CLEAR THE ESCAPE STATE
          3C A4    3A A1    A2    11CC  3005         SUBW    IRP$L_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); SUBTRACT OUT THE SEQUENCE
                                11D1  3006         SET_STATE <EOL,NOECHO>          ; LENGTH AND SET END OF LINE
                                11D9  3007
                     0047   31    11D9  3008 1830$:  BRW     1450$                   ; THEN EXIT
                                11DC  3009
          04 00000000'EF43    91    11DC  3010 1400$:  CMPB    TTY$A_CCLIST[R3],#TTY$K_ET_ESCAPE; CHECK FOR ESCAPE INTR
                          5C    12    11E4  3011         BNEQ    1470$                   ;ESCAPE SEQUENCE INTRODUCER
             53    7D 8F    91    11E6  3012         CMPB    #TTY$C_LOWESC1,R3       ;IS THIS AN ALTMODE
                   06    13    11EA  3013         BEQL    1410$                   ; YES THEN HANDLE SPECAILLY
             53    7E 8F    91    11EC  3014         CMPB    #TTY$C_LOWESC2,R3       ; LOWER ALSO
                   11    12    11F0  3015         BNEQ    1420$                   ; NO THEN MUST BE AN ESC OR CSI
          4B 44 A5    07    E0    11F2  3016 1410$:  BBS     #TT$V_LOWER,UCB$L_DEVDEPEND(R5),1470$; LOWER CASE TERMINAL
             51    3C A4    3C    11F7  3017         MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE LOCATION OF THIS CHARACTER
           00 B441    1B    90    11FB  3018         MOVB    #TTY$C_ESCAPE,@TTY$L_RB_TXT(R4)[R1]; AND FORCE THE ESCAPE
                53    1B    9A    1200  3019         MOVZBL  #TTY$C_ESCAPE,R3        ; AND PUT THE ESCAPE IN R3
                   3C A4    B6    1203  3020 1420$:  INCW    TTY$W_RB_TXTOFF(R4)     ; ESCAPE INTRO THEN SAY SO.
                     51    DD    1206  3021         PUSHL   R1                      ; SAVE
                  F291    30    1208  3022         BSBW    ESCINIT
           0103 C5    51    90    120B  3023         MOVB    R1,UCB$B_TT_ESC(R5)
                     51 8ED0    1210  3024         POPL    R1
                                1213  3025         SET_STATE ESC                   ; SETUP ESCAPE STATE
             51    58 A5    D0    1218  3026 1440$:  MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
             38 A1    53    B0    121C  3027         MOVW    R3,IRP$L_MEDIA(R1)      ; SETUP THE TERMINATING CHARACTER
                3A A1    B6    1220  3028         INCW    IRP$L_MEDIA+2(R1)       ; AND THE LENGTH
          40 A4    3C A4    B1    1223  3029 1450$:  CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); ARE WE FINSIHED
                   15    1E    1228  3030         BGEQU   1460$                   ; YES THEN FINISH
                                122A  3031         IF_NOT_STATE EOL,1480$          ; IF NOT END OF LINE THEN RETURN
          12 04 A2    1A    E4    122E  3032         BBSC    #TTY$V_ST_PRE,4(R2),1490$
                                1233  3033         IF_STATE NOECHO,1480$
                     0C    E0    1237  3034         BBS     #TRM$V_TM_TRMNOECHO,-
             0C 20 A4           1239  3035                 TTY$L_RB_MOD(R4),1495$  ; IF NOT ECHOING TERMINATER THEN EXIT
                  FF10    31    123C  3036 1480$:  BRW     1030$
                  FF53    31    123F  3037 1460$:  BRW     1600$
                  FEFB    31    1242  3038 1470$:  BRW     1040$
                  FF0F    31    1245  3039 1490$:  BRW     1020$
                  FF26    31    1248  3040 1495$:  BRW     1026$
```

```
                        124B  3042          .SBTTL  READ WITH VERIFICATION
                        124B  3043 ;++
                        124B  3044 ; RDVERIFY
                        124B  3045 ;
                        124B  3046 ; Description:
                        124B  3047 ;       Read verify allows programs that wish to do character validataion
                        124B  3048 ; to issue one IO rather than a QIO per character as was previously the case.
                        124B  3049 ;
                        124B  3050 ; Inputs:
                        124B  3051 ;
                        124B  3052 ;       R2 - Unit state vector
                        124B  3053 ;       R5 - address of the ucb
                        124B  3054 ;
                        124B  3055 ; Implicit inputs:
                        124B  3056 ;
                        124B  3057 ;       UCB$L_SVAPTE    - The address of the read buffer.
                        124B  3058 ;       TTY$L_RB_TXT    - The address of the first character of the initial string
                        124B  3059 ;       TTY$W_RB_TXTOFF - Offset to the last character in the initial string
                        124B  3060 ;       TTY$W_RB_TXTSIZ - Length of the data buffer.
                        124B  3061 ;       TTY$W_RB_LINOFF - Offset to the end of the field, initial offset.
                        124B  3062 ;       TTY$L_RB_PIC    - The address of the picture string.
                        124B  3063 ;--
                        124B  3064 RDVERIFY::
      54  00E4 C5   D0  124B  3065          MOVL    UCB$L_TT_TYPAHD(R5),R4  ; ADDRESS TYPEAHEAD BUFFER
                  25  13  1250  3066          BEQL    40$                     ; NO BUFFER THEN THERE IS A FORK
                        1252  3067                                          ; PROCESS WAITING SO LET IT GO.
                   0C A4  B5  1252  3068          TSTW    TTY$W_TA_INAHD(R4)      ; ANY TYPEAHEAD DATA?
                  25  12  1255  3069          BNEQ    50$                     ; yes then take it out and continue
                        1257  3070 ;
                        1257  3071 ; TYPEAHEAD BUFFER EXHAUSTED
                        1257  3072 ;
                        1257  3073          IF_NOT_STATE <TYPFUL>,35$       ; BR IF TYPEAHEAD IS NOT FULL
                        125B  3074
                 F3D4   31  125B  3075 20$:     BRW     XON                     ; CONTINUE
                        125E  3076 35$:
                        125E  3077 ;
                        125E  3078 ; THE TYPEAHEAD BUFFER HAS NOW BEEN EMPTIED INTO THE SYSTEM BUFFER.
                        125E  3079 ; IF PROCESSING A READ WITH ZERO SECOND TIMEOUT, RETURN THE DATA
                        125E  3080 ; TO THE USER IMMEDIATELY.
                        125E  3081 ;
          14 68 A5   01 E1  125E  3082          BBC     #UCB$V_TT_TIMO,UCB$W_DEVSTS(R5),40$; BR IF NOT READ WITH TIMEOUT.
00B0 C5   00000000'EF   D1  1263  3083          CMPL    TTY$A_MAXTIME,UCB$L_TT_RDUE(R5); ZERO SECOND TIMEOUT?
                  09  12  126C  3084          BNEQ    40$                     ; BR IF NOT.
        7C A5  022C 8F   B0  126E  3085          MOVW    #SS$_TIMEOUT,UCB$W_BOFF(R5); SET TIMEOUT COMPLETION STATUS.
                ED89'   30  1274  3086          BSBW    TTY$READONE             ; COMPLETE REQUEST.
                        1277  3087 ;
                        1277  3088 ; DISMISS INTERRUPT
                        1277  3089 ;
              010B C5   94  1277  3090 40$:     CLRB    UCB$B_TT_OUTYPE(R5)     ; SET NO RETURN CHARACTER.
                  05  127B  3091          RSB                             ; and dismiss the interupt
                        127C  3092 ;
                        127C  3093 ; ACTUALLY REMOVE CHARACTER FROM BUFFER
                        127C  3094 ;
          53   04 B4   9A  127C  3095 50$:     MOVZBL  @TTY$L_TA_GET(R4),R3    ; GET THE CHARACTER
       0F 04 A2   19  E5  1280  3096          BBCC    #TTY$V_ST_ECHAES,4(R2),55$; IS THIS THE FIRST CHARACTER
          54   78 A5   D0  1285  3097          MOVL    UCB$L_SVAPTE(R5),R4     ; GET THE READ PACKET ADDRESS
          44 A4   09   9B  1289  3098          MOVZBW  #TTY$K_ER_AESECHO,TTY$W_RE_MODE(R4); YES THEN ECHO THE AES STRING
```

```
                     128D  3099          SET_STATE EDITREAD                ; NOW DO EDITING
          FAB7   31  1291  3100          BRW     EDITREAD                  ; AND ECHO THE STRING
                     1294  3101 55$:
       0C A4   B7    1294  3102          DECW    TTY$W_TA_INAHD(R4)        ; ADJUST NUMBER OF TYPEAHEAD CHARACTERS
                     1297  3103 ;
                     1297  3104 ; CHECK FOR BUFFER WRAP AROUND
                     1297  3105 ;
06 04 A4   10 A4   F2  1297  3106          AOBLSS  TTY$L_TA_END(R4),TTY$L_TA_GET(R4),60$; POINTER PAST END?
 04 A4   0118 C4   9E  129D  3107          MOVAB   TTY$L_TA_DATA(R4),TTY$L_TA_GET(R4); RESET POINTER
                     12A3  3108
          50   DD  12A3  3109 60$:    PUSHL   R0                        ; charo can not destroy R0
    54   78 A5   D0  12A5  3110          MOVL    UCB$L_SVAPTE(R5),R4               ; GET READ BUFFER
                     12A9  3111          IF_STATE ESC,TO$                  ; IF ESCAPE SEQUENCE IN PROGRESS THEN HANDLE
    20 A4   11   E1  12AD  3112          BBC     #TRM$V_TM_R_JUST,TTY$L_RB_MOD(R4),-; BRANCH IF LEFT JUSTIFY MODE
                06   12B1  3113                  LEFT_JUSTIFY
              015C   31  12B2  3114          BRW     RIGHT_JUSTIFY
              029C   31  12B5  3115 10$:    BRW     ESCAPE_TERM
```

```
                                12B8  3117 ;
                                12B8  3118 ; THIS ROUTINE HANDLES LEFT JUSTIFIED FIELDS
                                12B8  3119 ;
                                12B8  3120
                                12B8  3121 LEFT_JUSTIFY:
                                12B8  3122
          51    30 A4    3C    12B8  3123          MOVZWL  TTY$W_RB_LINOFF(R4),R1          ; GET CURRENT OFFSET INTO FIELD
                                12BC  3124 ;
                                12BC  3125 ;        TERMINATE IF ALREADY FIELD FULL - THIS CAN OCCUR ON A NON AUTOTAB
                                12BC  3126 ;        FIELD WHERE WE DEFER "FULL FIELD" COMPLETION UNTIL ONE MORE
                                12BC  3127 ;        CHARACTER IS TYPED. CHAR TYPED IS TERMINATOR.
                                12BC  3128 ;
       3C A4      51    B1    12BC  3129          CMPW    R1,TTY$W_RB_TXTOFF(R4)          ; FIELD FULL?
                 03    19    12C0  3130          BLSS    5$                              ; IF LSS, NOT YET
               00A1    31    12C2  3131          BRW     TERM                            ; YES - INPUT CHAR IS TERM
                                12C5  3132 5$:
                                12C5  3133
                                12C5  3134 ; Check to see if this character is valid in this position
                                12C5  3135
         50    18 B441    9A    12C5  3136          MOVZBL  @TTY$L_RB_PIC(R4)[R1],R0        ; ADDRESS CORRESPONDING PIC VALUE
00000000'EF43    50    93    12CA  3137          BITB    R0,VERIFY_ARRAY[R3]             ; CHAR VALID ?
                 03    12    12D2  3138          BNEQ    10$                             ; YES,
               008F    31    12D4  3139          BRW     TERM                            ; ELSE TERMINATOR
                                12D7  3140 10$:
                                12D7  3141 ;
                                12D7  3142 ;        UPCASE IF REQUESTED.
                                12D7  3143 ;
       0A 20 A4      08    E1    12D7  3144          BBC     #TRM$V_TM_CVTLOW,TTY$L_RB_MOD(R4),13$; NO CONVERT THEN DON'T
 03 0000'CF43      03    E1    12DC  3145          BBC     #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],13$; is this lower case?
             53    20    8A    12E3  3146          BICB    #^X20,R3                        ; yes MAKE LOWERCASE UPPER
                                12E6  3147 13$:
          00 B441      53    90    12E6  3148          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]        ; STORE INTO READ BUFFER
                                12EB  3149 20$:
                 51    B6    12EB  3150          INCW    R1                              ; POINT TO NEXT POSITION
       3C A4      51    B1    12ED  3151          CMPW    R1,TTY$W_RB_TXTOFF(R4)          ; FIELD FULL?
                 5D    18    12F1  3152          BGEQ    FULL_1                          ; YES, WITH SINGLE CHARACTER
                                12F3  3153
                                12F3  3154 ; Check for trailing marker characters to be skipped
                                12F3  3155
          18 B441      95    12F3  3156          TSTB    @TTY$L_RB_PIC(R4)[R1]           ; NEXT POSITION MARKER?
                 39    12    12F7  3157          BNEQ    OUTPUT_1                        ; NO, OUTPUT 1 CHARACTER
                                12F9  3158
                 51    D7    12F9  3159          DECL    R1                              ; POINT TO PREVIOUS POSITION
    14 A4      00 B441      9E    12FB  3160          MOVAB   @TTY$L_RB_TXT(R4)[R1],-         ; SAVE ADDRESS OF STRING TO OUTPUT
                                1301  3161                  TTY$L_RB_ECHSTR(R4)
          50      01      9A    1301  3162          MOVZBL  #1,R0                           ; INIT COUNT OF CHARACTERS IN STRING
                                1304  3163 30$:
                 51    D6    1304  3164          INCL    R1                              ; POINT TO NEXT POSITION
       3C A4      51    B1    1306  3165          CMPW    R1,TTY$W_RB_TXTOFF(R4)          ; FIELD FULL
                 03    19    130A  3166          BLSS    35$                             ; NO
               0041    31    130C  3167          BRW     FULL_1                          ; YES, SO DON'T OUTPUT
                                130F  3168                                                  ; TRAILING MARKERS
                                130F  3169 35$:
          18 B441      95    130F  3170          TSTB    @TTY$L_RB_PIC(R4)[R1]           ; NEXT SLOT A MARKER
                 04    12    1313  3171          BNEQ    45$                             ; NO
                 50    B6    1315  3172          INCW    R0                              ; COUNT MARKERS
                 EB    11    1317  3173          BRB     30$                             ; CONTINUE LOOKING
```

```
                        1319  3174 45$:
0B A4   50   90         1319  3175          MOVB    R0,TTY$B_RB_ECHLEN(R4)          ; SAVE STRING LENGTH
                        131D  3176 ;         BRW     OUTPUT_S                        ; OUTPUT STRING
                        131D  3177
                        131D  3178 OUTPUT_S:                                         ; OUTPUT A STRING
30 A4   51   B0         131D  3179          MOVW    R1,TTY$W_RB_LINOFF(R4)          ; SAVE NEW OFFSET
                        1321  3180          IF_STATE NOECHO,DONE                    ; NOECHO THEN DON'T ECHO
   32 A4   B4           1325  3181          CLRW    TTY$W_RB_LINREST(R4)            ; DON'T CHANGE CURSOR POSITION
44 A4   0A   9B         1328  3182          MOVZBW  #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); SETUP TO ECHO
                        132C  3183          SET_STATE EDITREAD                      ;...
      18   11           1330  3184          BRB     DONE                            ; THEN ALLOW THE CODE TO FLOW
                        1332  3185
                        1332  3186 OUTPUT_1:                                       ; OUTPUT CHARACTER IN R3
30 A4   51   B0         1332  3187          MOVW    R1,TTY$W_RB_LINOFF(R4)          ; SAVE NEW OFFSET
47 A4   53   91         1336  3188          CMPB    R3,TTY$B_RB_RVFFIL(R4)          ; USER TYPE FILLCHAR
      04   12           133A  3189          BNEQ    EXIT                            ; NO
53   46 A4   90         133C  3190          MOVB    TTY$B_RB_RVFCLR(R4),R3          ; YES - REP W/CLR CHAR
                        1340  3191
                        1340  3192 EXIT:
                        1340  3193          IF_STATE NOECHO,DONE                    ; NO EDCHO THEN DON'T ECHO
   50 8ED0              1344  3194          POPL    R0
 EE3D   31              1347  3195          BRW     FORMAT_CHAR                     ; FORMAT THE CHARACTER ELSEWISE
                        134A  3196
                        134A  3197 DONE:
   50 8ED0              134A  3198          POPL    R0                              ; RESTORE R0
 ECB0   31              134D  3199          BRW     TTY$GETNEXTCHAR                 ; AND GET THE NEXT CHARACTER
                        1350  3200 FULL_1:                                         ; FIELD FULL, OUTPUT SINGLE CHARACTE
                        1350  3201 ;
                        1350  3202 ;       IF NOT AUTOTAB, DONT END NOW - LET HIM HIT A CHAR
                        1350  3203 ;
      12   E1           1350  3204          BBC     #TRM$V_TM_AUTO_TAB,-
   20 A4                1352  3205                  TTY$L_RB_MOD(R4),-
      DD                1354  3206                  OUTPUT_1                        ; ECHO LAST CHAR
                        1355  3207
30 A4   51   B0         1355  3208          MOVW    R1,TTY$W_RB_LINOFF(R4)          ; SAVE NEW OFFSET
                        1359  3209          SET_STATE EOL                           ; TELL THEM WE ARE DONE
54   58 A5   D0         135D  3210          MOVL    UCB$L_IRP(R5),R4                ; GET IRP ADDRESS
   38 A4   D4           1361  3211          CLRL    IRP$L_MEDIA(R4)                 ; SIGNAL NO TERMINATOR
      DA   11           1364  3212          BRB     EXIT
                        1366  3213
                        1366  3214 TERM:
30 A4   51   B0         1366  3215          MOVW    R1,TTY$W_RB_LINOFF(R4)          ; SAVE ENDING OFFSET
                        136A  3216          IF_STATE NOFLTR,3$                      ; SKIP EDITING IF NOFLTR
00000000'EF43   03   91 136E  3217          CMPB    #TTY$K_ET_DELEFT,TTY$A_CCLIST[R3]; IS THIS A DELETE CHARACTER
                46   13 1376  3218          BEQL    DELEFT_JUST                     ; YES THEN DELETE
                        1378  3219 3$:      IF_NOT_STATE ESCAPE,4$                  ; ESCAPE PROCESSING THEN
00000000'EF43   04   91 137C  3220          CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; ESCAPE ?
                24   13 1384  3221          BEQL    10$                             ; YES
                        1386  3222 4$:      SET_STATE EOL                           ; SET END OF LINE
30 A4   58 A5   D0      138A  3223 5$:      MOVL    UCB$L_IRP(R5),R0
   38 A0   53   9B      138E  3224          MOVZBW  R3,IRP$L_MEDIA(R0)              ; TERMINATOR
   3A A0   01   90      1392  3225          MOVB    #1,IRP$L_MEDIA+2(R0)            ; TERM LENGTH
   3B A0   51   90      1396  3226          MOVB    R1,IRP$L_MEDIA+3(R0)            ; OFFSET TO INVALID CHAR
   51   3C A4   3C      139A  3227          MOVZWL  TTY$W_RB_TXTOFF(R4),R1          ; GET THE OFFSET TO THE END
                        139E  3228                                                  ; OF THE BUFFER
18 A4   00 B441   9E    139E  3229          MOVAB   @TTY$L_RB_TXT(R4)[R1],TTY$L_RB_PIC(R4);SETUP THE END OF THE BUFFER
   18 B4   53   90      13A4  3230          MOVB    R3,@TTY$L_RB_PIC(R4)            ; STORE IN BUFFER
```

```
                A0   11   13A8  3231          BRB     DONE
                                 13AA  3232 ;
                                 13AA  3233 ; SIGNAL ESCAPE SEQUENCE ACTIVE
                                 13AA  3234 ;
                                 13AA  3235 10$:
                                 13AA  3236          SET_STATE ESC
                51   DD   13AF  3237          PUSHL   R1                      ; SAVE R1
              FOE8   30   13B1  3238          BSBW    ESCINIT                 ; INIT THE ESCAPE SEQUENCE RULES
0103 C5         51   90   13B4  3239          MOVB    R1,UCB$B_TT_ESC(R5)     ; ...
                51 8ED0   13B9  3240          POPL    R1                      ; RESTORE R1
                                 13BC  3241
                CC   11   13BC  3242          BRB     5$                              ; SKIP SETTING END
                                 13BE  3243 ;
                                 13BE  3244 ; DELETE ONE CHARACTER TO THE LEFT
                                 13BE  3245 ;
                                 13BE  3246 ;       R1 = offset into the line
                                 13BE  3247 ;
                                 13BE  3248 DELEFT_JUST:
                50   D4   13BE  3249          CLRL    R0                      ; A COUNTER
                51   D7   13C0  3250          DECL    R1                      ; move left 1 character
                0C   19   13C2  3251          BLSS    100$                    ; NO CHARACTER TO DELETE THEN DON'T
                50   D6   13C4  3252 10$:     INCL    R0                      ; MOVE BACK 1 CHARACTER
          18 B441    95   13C6  3253          TSTB    @TTY$L_RB_PIC(R4)[R1]   ; IS THIS A MARKER?
                07   12   13CA  3254          BNEQ    20$                     ; NO THEN SHIFT
                51   D7   13CC  3255          DECL    R1                      ; ELSE MOVE OVER IT
                F4   18   13CE  3256          BGEQ    10$                     ; NO NON-MARKER THEN IGNORE THE DEL
              FF77   31   13D0  3257 100$:    BRW     DONE                    ; ...
                                 13D3  3258
3C A4     30 A4    B1   13D3  3259 20$:     CMPW    TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4); WERE WE ABOUT TO TERMINATE
                03   12   13D8  3260          BNEQ    25$                     ; NO THEN PROCESS NORMALY
             50   01   D0   13DA  3261          MOVL    #1,R0                   ; ELSE ONLY GO BACK 1 PLACE
                                 13DD  3262                                  ; DUE TO SPECAIL EOF CONDITIONS
          30 A4    51   B0   13DD  3263 25$:     MOVW    R1,TTY$W_RB_LINOFF(R4)  ; UPDATE THE OFFSET
                                 13E1  3264          IF_STATE NOECHO,50$
38 A4     00FC C5    50   A3   13E5  3265          SUBW3   R0,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); SET THE FINAL CURSOR
          0100 C5    50   B0   13EC  3266          MOVW    R0,UCB$W_TT_BSPLEN(R5)  ; AND TELL HOW MANY CHARACTERS TO
                                 13F1  3267                                  ; BACKSPACE OVER
                                 13F1  3268          SET_STATE <BACKSPACE,EDITREAD>  ; THEN OUTPUT OTHER STUFF AND RETURN
       14 A4    00 B441    9E   13F6  3269          MOVAB   @TTY$L_RB_TXT(R4)[R1],TTY$L_RB_ECHSTR(R4); GET THE ADDRESS OF WHERE
          0B A4    01   90   13FC  3270          MOVB    #1,TTY$B_RB_ECHLEN(R4)  ; THE LENGTH OF THE STRING TO OUTPUT
          44 A4    0A   9B   1400  3271          MOVZBW  #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); SET THE MODE OF THE EDITING
          32 A4    01   D0   1404  3272          MOVL    #1,TTY$W_RB_LINREST(R4) ; TELL THE MOVE TO RESTORE THE POSIT
                                 1408  3273
       00 B441    47 A4    90   1408  3274 50$:     MOVB    TTY$B_RB_RVFFIL(R4),@TTY$L_RB_TXT(R4)[R1]; AND FILL THE BLANK
                                 140E  3275                                  ; WITH THE FILL CHARACTER
                FF39   31   140E  3276          BRW     DONE
```

```
                                                C 12
                     1411  3278 RIGHT_JUSTIFY:
                     1411  3279
          51  30 A4   32  1411  3280          CVTWL   TTY$W_RB_LINOFF(R4),R1          ; GET CURRENT OFFSET
                03    18  1415  3281          BGEQ    3$                             ; IF WE ARE BEOND THE
                     1417  3282                                                       ; BOUNDARY THEN TERMINATE
              00BD    31  1417  3283          BRW     RIGHT_TERM
                     141A  3284 3$:
        50  18 B441   9A  141A  3285          MOVZBL  @TTY$L_RB_PIC(R4)[R1],R0       ; GET PICTURE VALUE
                0D    12  141F  3286          BNEQ    5$                             ; NOT A MARKER
                51    D7  1421  3287          DECL    R1                             ; CHECK FOR END
                F5    18  1423  3288          BGEQ    3$                             ; NOT YET
                51    D4  1425  3289          CLRL    R1
          30 A4   51    B0  1427  3290          MOVW    R1,TTY$W_RB_LINOFF(R4)         ; SAVE CURRENT OFFSET
              0076    31  142B  3291          BRW     FIELD_FULL                     ; FULL OF MARKERS
                     142E  3292 5$:
00000000'EF43   50    93  142E  3293          BITB    R0,VERIFY_ARRAY[R3]            ; IS THIS CHARACTER VALID HERE
                03    12  1436  3294          BNEQ    10$                            ; OK
              009C    31  1438  3295          BRW     RIGHT_TERM
                     143B  3296 ;
                     143B  3297 ; SHIFT DATA IN INPUT FIELD
                     143B  3298 ;
                     143B  3299 10$:
          30 A4   51    B0  143B  3300          MOVW    R1,TTY$W_RB_LINOFF(R4)         ; SAVE CURRENT OFFSET
                     143F  3301
                51    D4  143F  3302          CLRL    R1                             ; INIT DESTINATION POINTER
                     1441  3303
                     1441  3304 ; FIND FIRST VALID DESTINATION
                     1441  3305
                     1441  3306 20$:
        18 B441   95  1441  3307          TSTB    @TTY$L_RB_PIC(R4)[R1]          ; MARKER?
                0B    12  1445  3308          BNEQ    30$                            ; NO, POSITION IS OK
                51    D6  1447  3309          INCL    R1                             ; LOOK FOR NEW DESTINATION
          3C A4   51    B1  1449  3310          CMPW    R1,TTY$W_RB_TXTOFF(R4)         ; PAST END?
                F2    1F  144D  3311          BLSSU   20$                            ; NO, IT IS STILL OK
              0052    31  144F  3312          BRW     FIELD_FULL
                     1452  3313 30$:
                     1452  3314 ;
                     1452  3315 ; IF NONCLR CHAR IN LEFTMOST DATA POSITION, FIELD IS ALREADY FULL, MUST
                     1452  3316 ;       BE NON-AUTOTAB. TERMINATE WITH OVERFLOWING CHAR
                     1452  3317 ;
47 A4   00 B441   91  1452  3318          CMPB    @TTY$L_RB_TXT(R4)[R1],TTY$B_RB_RVFFIL(R4);THIS A CLR CHAR?
                07    13  1458  3319          BEQL    35$                            ;YES - NOT FULL YET
             3C A4    B0  145A  3320          MOVW    TTY$W_RB_TXTOFF(R4),-          ;SHOW OVERFLOW OFFSET
             30 A4        145D  3321                  TTY$W_RB_LINOFF(R4)            ;IF FIELD FULL
                76    11  145F  3322          BRB     RIGHT_TERM                     ; AND QUIT
                     1461  3323 35$:
                51    DD  1461  3324          PUSHL   R1                             ;SAVE OFFSET TO LEFT BYTE
                     1463  3325
          50    51    D0  1463  3326          MOVL    R1,R0                          ; INIT SOURCE POINTER
                     1466  3327
                     1466  3328 ; FIND NEW SOURCE
                     1466  3329
                     1466  3330 40$:
                50    D6  1466  3331          INCL    R0
          30 A4   50    B1  1468  3332          CMPW    R0,TTY$W_RB_LINOFF(R4)         ; PAST END?
                12    1A  146C  3333          BGTRU   100$                           ; YES THEN SHIFT COMPLETE
             18 B440   95  146E  3334          TSTB    @TTY$L_RB_PIC(R4)[R0]          ; MARKER?
```

```
                F2   13  1472  3335          BEQL    40$                             ; YES, LOOK AGAIN
     00 B441   00 B440   90  1474  3336          MOVB    @TTY$L_RB_TXT(R4)[R0],@TTY$L_RB_TXT(R4)[R1]; SHIFT THE DATA
               51   50   D0  147B  3337          MOVL    R0,R1                           ; OLD SOURCE IS NEW DESTINATION
                    E6   11  147E  3338          BRB     40$
                         1480  3339
                         1480  3340 ; SHIFT COMPLETE
                         1480  3341
                         1480  3342 100$:
                         1480  3343 ;
                         1480  3344 ;       UPCASE IF REQUESTED.
                         1480  3345 ;
                    08   E1  1480  3346          BBC     #TRM$V_TM_CVTLOW,-              ; SKIP IF UPCASE NOT
                  0A 20 A4        1482  3347                  TTY$L_RB_MOD(R4),105$       ;       REQUESTED
     03 0000'CF43   03   E1  1485  3348          BBC     #TTY$V_CR_LOWER,W^TTY$A_TYPE[R3],105$; is this lower case?
               53   20   8A  148C  3349          BICB    #^X20,R3                        ; yes MAKE LOWERCASE UPPER
                         148F  3350 105$:
                         148F  3351
          00 B441   53   90  148F  3352          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]        ; STORE NEW CHARACTER INTO
                         1494  3353                                                  ; END POSITION
                    51 8ED0  1494  3354          POPL    R1                              ; GET START POSITION
        46 A4   00 B441   91  1497  3355          CMPB    @TTY$L_RB_TXT(R4)[R1],TTY$B_RB_RVFCLR(R4);START WITH CLR CHAR?
                    10   13  149D  3356          BEQL    RIGHT_ECHO                      ;YES - FLD NOT FULL
                    12   E1  149F  3357          BBC     #TRM$V_TM_AUTO_TAB,-            ;FLD FULL - NONAUTOTAB
                  0B 20 A4        14A1  3358                  TTY$L_RB_MOD(R4),RIGHT_ECHO ; JUST ECHO
                         14A4  3359                                                  ;FLD FULL - AUTOTAB
                         14A4  3360                                                  ; FALL THROUGH
                         14A4  3361
                         14A4  3362
                         14A4  3363 FIELD_FULL:
                         14A4  3364          SET_STATE EOL
                         14A8  3365 10$:
             51   58 A5   D0  14A8  3366          MOVL    UCB$L_IRP(R5),R1                ; GET IRP ADDRESS
                  38 A1   D4  14AC  3367          CLRL    IRP$L_MEDIA(R1)                 ; SIGNAL NO TERMINATOR
                         14AF  3368
                         14AF  3369 RIGHT_ECHO:
                00FC C5   B4  14AF  3370          CLRW    UCB$W_TT_CURSOR(R5)
                         14B3  3371          IF_STATE NOECHO,GODONE                  ; NO ECHO THEN DON'T ECHO
                    50 8ED0  14B7  3372          POPL    R0
                  32 A4   B4  14BA  3373          CLRW    TTY$W_RB_LINREST(R4)
           44 A4    0A   9B  14BD  3374          MOVZBW  #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); ECHO THE LINE
        14 A4   4A A4   9E  14C1  3375          MOVAB   TTY$A_RB_PRM(R4),TTY$L_RB_ECHSTR(R4); THE ADDRESS OF THE STRING
        30 A4   34 A4   81  14C6  3376          ADDB3   TTY$W_RB_PRMLEN(R4),TTY$W_RB_LINOFF(R4),-; GET THE LENGTH TO ECHO
                  0B A4         14CB  3377                  TTY$B_RB_ECHLEN(R4)         ;...
                  0B A4   96  14CD  3378          INCB    TTY$B_RB_ECHLEN(R4)             ; ACCOUNT FOR ZERO BASE
                         14D0  3379          SET_STATE EDITREAD
                   F874   31  14D4  3380          BRW     EDITREAD
                         14D7  3381
                         14D7  3382 .ENABLE LSB
                         14D7  3383
                         14D7  3384 RIGHT_TERM:
                         14D7  3385          IF_STATE NOFLTR,3$                      ; SKIP EDITING IF NOFLTR
00000000'EF43   03   91  14DB  3386          CMPB    #TTY$K_ET_DELEFT,TTY$A_CCLIST[R3]; IS THIS A DELETE CHARACTER
                4D   13  14E3  3387          BEQL    DELRIGHT_JUST                   ; YES THEN DELETE
                         14E5  3388 3$:       IF_NOT_STATE ESCAPE,4$                  ; ESCAPE PROCESSING THEN
00000000'EF43   04   91  14E9  3389          CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; ESCAPE ?
                2B   13  14F1  3390          BEQL    10$                             ; YES
     7E A5   3C A4   B0  14F3  3391 4$:       MOVW    TTY$W_RB_TXTOFF(R4),UCB$W_BCNT(R5); FILL REQUIRED FIELD
```

```
                        14F8  3392          SET_STATE <EOL>                         ; SET END OF LINE
                        14FC  3393 5$:
          51   58 A5    D0  14FC  3394          MOVL    UCB$L_IRP(R5),R1            ; GET IRP ADDRESS
        38 A1    53    9B  1500  3395          MOVZBW  R3,IRP$L_MEDIA(R1)          ; TERMINATOR
        3A A1    01    90  1504  3396          MOVB    #1,IRP$L_MEDIA+2(R1)        ; TERM LENGTH
              30 A4    90  1508  3397          MOVB    TTY$W_RB_LINOFF(R4),-
                 3B A1        150B  3398                  IRP$L_MEDIA+3(R1)       ; OFFSET TO INVALID CHAR
          51   3C A4    3C  150D  3399          MOVZWL  TTY$W_RB_TXTOFF(R4),R1      ; GET THE ADDRESS OF THE END
    18 A4    00 B441    9E  1511  3400          MOVAB   @TTY$[_RB_TXT(R4)[R1],TTY$L_RB_PIC(R4);OF THE INITIAL STRING
        18 B4    53    90  1517  3401          MOVB    R3,@TTY$L_RB_PIC(R4)        ; AND STORE THE CHARACTER THERE
              FE2C    31  151B  3402 GODONE:  BRW     DONE                        ; THEN EXIT
                        151E  3403 ;
                        151E  3404 ; SIGNAL ESCAPE SEQUENCE ACTIVE
                        151E  3405 ;
                        151E  3406 10$:
                        151E  3407          SET_STATE ESC                           ; ANNOUNCE ESCAPE STATE
                51    DD  1523  3408          PUSHL   R1                      ; SAVE R1
              EF74    30  1525  3409          BSBW    ESCINIT                 ; INIT THE ESCAPE SEQUENCE RULES
    0103 C5     51    90  1528  3410          MOVB    R1,UCB$B_TT_ESC(R5)     ;
                51 8ED0  152D  3411          POPL    R1                      ; RESTORE R1
                        1530  3412
                CA    11  1530  3413          BRB     5$                              ; SKIP SETTING END
                        1532  3414
                        1532  3415          .DISABLE LSB
                        1532  3416 ;
                        1532  3417 ; DELETE RIGHT JUSTIFYED
                        1532  3418 ;
                        1532  3419 DELRIGHT_JUST:
             50   51    D0  1532  3420          MOVL    R1,R0                   ; GET THE CURRENT LOCATION
                  50    D7  1535  3421 10$:     DECL    R0                      ; MOVE RIGHT 1 CHARACTER
                  12    19  1537  3422          BLSS    20$                     ; ECHO THE LINE
                18 B440 95  1539  3423          TSTB    @TTY$L_RB_PIC(R4)[R0]   ; IS THIS A MARKER?
                    F6  13  153D  3424          BEQL    10$                     ; YES THEN SKIP IT
    00 B441     00 B440 90  153F  3425          MOVB    @TTY$L_RB_TXT(R4)[R0],- ; MOVE THE CHARACTER
                        1546  3426                  @TTY$L_RB_TXT(R4)[R1]
                51   50 D0  1546  3427          MOVL    R0,R1
                    EA  11  1549  3428          BRB     10$
                        154B  3429
                  47 A4 90  154B  3430 20$:     MOVB    TTY$B_RB_RVFFIL(R4),-   ; MOVE A FILL CHARACTER FROM THE LEFT
                00 B441     154E  3431                  @TTY$[_RB_TXT(R4)[R1]
                    FF5B  31  1551  3432          BRW     RIGHT_ECHO              ; NOW ECHO THE CHARACTER
```

```
                     1554  3434          .SBTTL  READ VERIFY ESCAPE TERMINATOR ROUTINE
                     1554  3435 ;++
                     1554  3436 ; DESCRIPTION:
                     1554  3437 ;       CALLED ON ALL THE CHARACTERS FOLLOWING THE ESCAPE
                     1554  3438 ;
                     1554  3439 ; INPUTS:
                     1554  3440 ;
                     1554  3441 ;       R2 - UNIT STATE VECTOR
                     1554  3442 ;       R4 - ADDRESS OF THE READ BUFFER
                     1554  3443 ;       R5 - UCB ADDRESS
                     1554  3444 ;
                     1554  3445 ; IMPLICIT INPUTS:
                     1554  3446 ;
                     1554  3447 ;       IRP$L_MEDIA+2 - CURRENT OFFSET TO IN THE BUFFER (ACCESSED AS A BYTE)
                     1554  3448 ;       TTY$L_RB_PIC  - THE ADDRESS OF THE ESCAPE CHARACTER IN THE BUFFER
                     1554  3449 ;       TTY$W_RB_TXTSIZ - LENGTH OF THE USERS BUFFER
                     1554  3450 ;--
                     1554  3451 ESCAPE_TERM:
 50   58 A5    D0     1554  3452          MOVL    UCB$L_IRP(R5),R0                ; GET IRP ADDRESS
 51   3A A0    9A     1558  3453          MOVZBL  IRP$L_MEDIA+2(R0),R1            ; GET CURRENT OFFSET
                     155C  3454
 40 A4    51    B1     155C  3455          CMPW    R1,TTY$W_RB_TXTSIZ(R4)          ; DO WE HAVE ROOM FOR THIS
                14    1E     1560  3456          BGEQU   30$                     ;NO THEN TELL THEM ABOUT IT
18 B441    53    90     1562  3457          MOVB    R3,@TTY$L_RB_PIC(R4)[R1]        ; STORE THE DATA
      3A A0    96     1567  3458          INCB    IRP$L_MEDIA+2(R0)               ; BUMP TERM LENGTH
                     156A  3459
       EF40    30     156A  3460          BSBW    ESCSYNTAX                       ; PARSE ESCAPE SEQUENCE
         0B    14     156D  3461          BGTR    10$                             ; SEQUENCE OK, CONTINUE
         0C    19     156F  3462          BLSS    20$                             ; SEQUENCE ERROR
                     1571  3463
                     1571  3464 ; SEQUENCE COMPLETED NORMALLY
                     1571  3465          CLR_STATE ESC                           ; CLEAR THE ESCAPE STATE THEN
                     1576  3466
                     1576  3467 30$:     SET_STATE EOL                           ; SET END OF LINE
                     157A  3468 10$:
FDCD    31           157A  3469          BRW     DONE
                     157D  3470
                     157D  3471 20$:                                             ; ESCAPE SEQ ERROR
                     157D  3472          SET_STATE BADESC                        ; TELL THE WORLD
  F3    11           1581  3473          BRB     30$
```

```
1583  3475          .SBTTL  End of module
1583  3476 TT_END:
1583  3477          .END
```

```
ADJUST_CURSOR                   000001A7 R      02
AESECHO                         00000DF5 R      02
BACKSPACING                     000003F4 R      02
BACK_CHAR                       0000088B R      02
BACK_TAB                        00000EEF R      02
BSPACE                          00000204 R      02
BUFEMPTY                        0000060B R      02
BURST                           00000099 R      02
CALLBACK                        00000DDC R      02
CARRIAGE                        0000020B R      02
CHAR_SIZE                       00000F52 R      02
CHECK_BUFFER                    00000593 R      02
CLRECHO                         00000E04 R      02
CLRREST                         00000E17 R      02
CNT                           = 00000001
COMECHO                         00000E0C R      02
CROUTPUT                        0000020F R      02
CTRLR                           000008C8 R      02
CTRLU                           00000908 R      02
CTRLZ                           0000029C R      02
CURSOROVRFLOW                   00000406 R      02
CVTLOW                          0000064D R      02
DELCHAR                         000009C3 R      02
DELEFT_JUST                     000013BE R      02
DELESCAPE                       00000B41 R      02
DELETE_WORD                     00000B5A R      02
DELRIGHT_JUST                   00001532 R      02
DISMISS                         0000065A R      02
DONE                            0000134A R      02
DROP                            00000299 R      02
DROP_CHAR                       00000B57 R      02
ECHLINE                         00000E44 R      02
EDITINGCHAR                     00000803 R      02
EDITREAD                        00000D4B R      02
EOLSEEN                         0000043B R      02
ESCAPE                          000002C5 R      02
ESCAPE_CHAR                     00000BF2 R      02
ESCAPE_TAB                      00000F92 R      02
ESCAPE_TERM                     00001554 R      02
ESCINIT                         0000049C RG     02
ESCINPROG                       0000065F R      02
ESCSYNTAX                       000004AD R      02
ESCSYNTAX_O                     000004C3 R      02
EXE$GL_ABSTIM                   ********    X   02
EXIT                            00001340 R      02
EXITING                         00000DC7 R      02
E_SYNTAX                        000004D0 RG     02
F                             = 00000000
FIELD_FULL                      000014A4 R      02
FILLING                         00000450 R      02
FIND_BOL                        00000FC1 R      02
FIND_BOL_NOCLEAR                00000FC5 RG     02
FORM                            000003C6 R      02
FORMAT                          000000D6 R      02
FORMAT_CHAR                     00000187 R      02
FORMAT_LOCAL                    000001E7 R      02
FORMAT_X                        0000013C R      02
FORWARD_CHAR                    00000C51 R      02
FULL_1                          00001350 R      02
GETNEXTCHAR                     00000005 R      02
GODEL                           00000B54 R      02
GODONE                          0000151B R      02
GODROP                          00000C89 R      02
GOOUT                           00000201 R      02
INDELST                         0000073B R      02
INSERT_CHAR                     00000571 R      02
INTEXIT                         0000003D R      02
INTEXIT1                        00000048 R      02
IO$M_TRMNOECHO                =  00001000
IO$V_CVTLOW                   =  00000008
IO$V_NOECHO                   =  00000006
IO$V_TRMNOECHO                =  0000000C
IRP$C_MEDIA                   =  00000038
IRP$W_FUNC                    =  00000020
LEFT_JUSTIFY                    000012B8 R      02
LFOUTPUT                        0000031A R      02
LINEFEED                        00000315 R      02
LOADADR                         0000107A R      02
MOVECURSOR                      00000E4A R      02
MOVEREADATA                     00000526 R      02
MOVE_BOL                        00000C8C R      02
MOVE_EOL                        00000CDB R      02
MOVIFNEC                        00000DBF R      02
MULTIECHOING                    0000045E R      02
NORMAL                          00000D6C R      02
NOTA                            00000619 R      02
OPEN                            00000752 R      02
OUTPUTANDWAIT                   00000148 R      02
OUTPUTANDWAIT1                  0000014D R      02
OUTPUT_1                        00001332 R      02
OUTPUT_S                        0000131D R      02
PASSALL                         0000112B RG     02
POWERREST                       00000042 R      02
PRMECHO                         00000E51 R      02
PRMECHO1                        00000E6E R      02
QUOTE                           000007EE R      02
QUOTING                         00000D06 R      02
RDVERIFY                        0000124B RG     02
RECALL                          00000D0E R      02
RECALLING                       0000085C R      02
RESTART                         ********    X   02
RIGHT_ECHO                      000014AF R      02
RIGHT_JUSTIFY                   00001411 R      02
RIGHT_TERM                      000014D7 R      02
RVECHO                          00000E97 R      02
SENDLINEFEED                    00000493 R      02
SIMCEOL                         00000EB9 R      02
SS$_CONTROLO                  =  00000609
SS$_DATAOVERUN                =  00000838
SS$_NORMAL                    =  00000001
SS$_TIMEOUT                   =  0000022C
STRTMULTI                       00001060 RG     02
STRTMULTI_1                     0000106E RG     02
STR_EXIT                        0000009E R      02
```

```
STR_TIMESET                     000000AF R      02
T                             = 00000028
TAB                             00000358 R      02
TABRIGHT                        0000108D R      02
TAB_BUFEMPTY                    00000507 R      02
TAB_CVTLOW                      00000520 R      02
TAB_DISMISS                     00000504 R      02
TAB_EDITINGCHAR                 00000516 R      02
TAB_EOLSEEN                     000004FA R      02
TAB_ESCINPROG                   00000519 R      02
TAB_GETNEXT                     00000510 R      02
TAB_INDELST                     0000050A R      02
TAB_LOCAL                       000003A5 R      02
TAB_NOTA                        000004F7 R      02
TAB_OPEN                        00000523 R      02
TAB_PASSALL                     0000050D R      02
TAB_QUOTE                       00000513 R      02
TERM                            00001366 R      02
TERMFOUND                       000005BC R      02
TOGGELINSOV                     00000D3E R      02
TRMSV_TM_AUTO_TAB             = 00000012
TRMSV_TM_CVTLOW               = 00000008
TRMSV_TM_NOECHO               = 00000006
TRMSV_TM_NORECALL             = 00000010
TRMSV_TM_R_JUST               = 00000011
TRMSV_TM_TRMNOECHO            = 0000000C
TT$V_LOWER                    = 00000007
TT$V_MECHFORM                 = 00000013
TT$V_MECHTAB                  = 00000008
TT$V_SCOPE                    = 0000000C
TT$V_WRAP                     = 00000009
TT$_VT100                     = 00000060
TT$_VT5X                      = 00000040
TT2$V_ANSICRT                 = 00000018
TT2$V_DECCRT                  = 0000001D
TT2$V_EDITING                 = 0000000C
TT2$V_LOCALECHO               = 00000000
TTY$A_ANSIBACKUP                ******** X      02
TTY$A_ANSICEL                   ******** X      02
TTY$A_ANSI_DEOL                 ******** X      02
TTY$A_ANSI_UPCEL                ******** X      02
TTY$A_CCLIST                    ******** X      02
TTY$A_CTRLR                     ******** X      02
TTY$A_CTRLU                     ******** X      02
TTY$A_DELCRTTAB                 ******** X      02
TTY$A_ESCAPE                    ******** X      02
TTY$A_ESCINIT                   ******** X      02
TTY$A_EXITECHO                  ******** X      02
TTY$A_FCNTKN                    ******** X      02
TTY$A_FORM                      ******** X      02
TTY$A_LONGFORM                  ******** X      02
TTY$A_MAXTIME                   ******** X      02
TTY$A_PREFIX                    ******** X      02
TTY$A_RB_PRM                  = 0000004A
TTY$A_SPACEBACK                 ******** X      02
TTY$A_TAB                       ******** X      02
TTY$A_TA_RCL                  = 00000018
TTY$A_TYPE                      ******** X      02
TTY$A_VTAB                      ******** X      02
TTY$A_WORDTERM                  ******** X      02
TTY$B_RB_ECHLEN               = 0000000B
TTY$B_RB_RVFCLR               = 00000046
TTY$B_RB_RVFFIL               = 00000047
TTY$C_BLANK                   = 00000020
TTY$C_BS                      = 00000008
TTY$C_CR                      = 0000000D
TTY$C_CSI                     = 0000009B
TTY$C_CTRLB                   = 00000002
TTY$C_CTRLZ                   = 0000001A
TTY$C_DELETE                  = 0000007F
TTY$C_DOLLAR                  = 00000024
TTY$C_ESCAPE                  = 0000001B
TTY$C_LF                      = 0000000A
TTY$C_LOWESC1                 = 0000007D
TTY$C_LOWESC2                 = 0000007E
TTY$C_TAB                     = 00000009
TTY$GETNEXTCHAR                 00000000 RG     02
TTY$K_EDITNORMAL              = 00000004
TTY$K_ER_AESECHO              = 00000009
TTY$K_ER_CLRECHO              = 00000001
TTY$K_ER_CLRREST              = 00000006
TTY$K_ER_ECHLINE              = 00000002
TTY$K_ER_EXITING              = 00000004
TTY$K_ER_MOVECURSOR           = 00000005
TTY$K_ER_PRMECHO              = 00000007
TTY$K_ER_PRMECHO1             = 00000008
TTY$K_ER_RVECHO               = 0000000A
TTY$K_ER_SIMCEOL              = 0000000B
TTY$K_ET_BACK_CHAR            = 00000005
TTY$K_ET_DELEFT               = 00000003
TTY$K_ET_ESCAPE               = 00000004
TTY$K_ET_FORWARD_CHAR         = 00000006
TTY$K_ET_RECALL               = 0000000B
TTY$K_ET_TERMINATE            = 0000000E
TTY$K_ET_UNUSED               = 000C000D
TTY$K_MAXESCTKN                 ******** X      02
TTY$L_RB_AES                  = 00000024
TTY$L_RB_DATA                 = 0000004A
TTY$L_RB_ECHSTR               = 00000014
TTY$L_RB_LIN                  = 0000002C
TTY$L_RB_MOD                  = 00000020
TTY$L_RB_PIC                  = 00000018
TTY$L_RB_TERM                 = 0000001C
TTY$L_RB_TXT                  = 00000000
TTY$L_TA_DATA                 = 00000118
TTY$L_TA_END                  = 00000010
TTY$L_TA_GET                  = 00000004
TTY$L_WB_END                  = 00000020
TTY$L_WB_IRP                  = 00000024
TTY$L_WB_NEXT                 = 0000001C
TTY$M_CH_CTRL                 = 00000020
TTY$M_CH_CTRL2                = 00000080
TTY$M_CH_CTRL3                = 00000040
TTY$M_CH_SPEC                 = 00000010
```

```
TTY$M_ST_BACKSPACE              = 00000020
TTY$M_ST_BADESC                 = 00000100
TTY$M_ST_CTRLO                  = 00000001
TTY$M_ST_CTRLR                  = 00040000
TTY$M_ST_CURSOR                 = 00000008
TTY$M_ST_DEL                    = 00000002
TTY$M_ST_EDITING                = 00100000
TTY$M_ST_EDITREAD               = 00000200
TTY$M_ST_EOL                    = 00000100
TTY$M_ST_ESC                    = 00000080
TTY$M_ST_ESCAPE                 = 00000800
TTY$M_ST_ESC_O                  = 00004000
TTY$M_ST_FILL                   = 00000004
TTY$M_ST_MULTI                  = 00000040
TTY$M_ST_NINTMULTI              = 08000000
TTY$M_ST_NL                     = 00000200
TTY$M_ST_NOECHO                 = 00000008
TTY$M_ST_NOFLTR                 = 00000040
TTY$M_ST_OVERSTRIKE             = 00800000
TTY$M_ST_OVRFLO                 = 00010000
TTY$M_ST_PASALL                 = 00000004
TTY$M_ST_PRE                    = 04000000
TTY$M_ST_QUOTING                = 00400000
TTY$M_ST_RDVERIFY               = 00000400
TTY$M_ST_READ                   = 00001000
TTY$M_ST_RECALL                 = 00000800
TTY$M_ST_SENDLF                 = 00000010
TTY$M_ST_SKIPCRLF               = 00080000
TTY$M_ST_SKIPLF                 = 00002000
TTY$M_ST_TABEXPAND              = 00200000
TTY$M_ST_TABRIGHT               = 40000000
TTY$M_ST_TYPFUL                 = 00001000
TTY$M_ST_WRAP                   = 00008000
TTY$M_ST_WRITE                  = 00000080
TTY$M_ST_WRTALL                 = 00000010
TTY$Q_RB_ECHOAREA               = 0000000C
TTY$READONE                     ******** X   02
TTY$V_CH_LOWER                  = 00000003
TTY$V_CH_SPEC                   = 00000004
TTY$V_PC_NOCRLF                 = 00000007
TTY$V_PC_NOTIME                 = 00000000
TTY$V_RS_WRAP                   = 00000000
TTY$V_ST_DEL                    = 00000001
TTY$V_ST_ECHAES                 = 00000019
TTY$V_ST_EDITING                = 00000014
TTY$V_ST_ESC_O                  = 0000000E
TTY$V_ST_OVERSTRIKE             = 00000017
TTY$V_ST_PRE                    = 0000001A
TTY$V_ST_READ                   = 0000000C
TTY$V_ST_SKIPLF                 = 0000000D
TTY$V_SX_BACKSPACE              = 00000005
TTY$V_SX_BADESC                 = 00000028
TTY$V_SX_CTRLO                  = 00000020
TTY$V_SX_CTRLR                  = 00000032
TTY$V_SX_CURSOR                 = 00000003
TTY$V_SX_DEL                    = 00000021
TTY$V_SX_EDITING                = 00000034
TTY$V_SX_EDITREAD               = 00000009
TTY$V_SX_EOL                    = 00000008
TTY$V_SX_ESC                    = 00000027
TTY$V_SX_ESCAPE                 = 0000002B
TTY$V_SX_ESC_O                  = 0000002E
TTY$V_SX_FILL                   = 00000002
TTY$V_SX_MULTI                  = 00000006
TTY$V_SX_NINTMULTI              = 0000003B
TTY$V_SX_NL                     = 00000029
TTY$V_SX_NOECHO                 = 00000023
TTY$V_SX_NOFLTR                 = 00000026
TTY$V_SX_OVERSTRIKE             = 00000037
TTY$V_SX_OVRFLO                 = 00000030
TTY$V_SX_PASALL                 = 00000022
TTY$V_SX_PRE                    = 0000003A
TTY$V_SX_QUOTING                = 00000036
TTY$V_SX_RDVERIFY               = 0000000A
TTY$V_SX_READ                   = 0000000C
TTY$V_SX_RECALL                 = 0000000B
TTY$V_SX_SENDLF                 = 00000004
TTY$V_SX_SKIPCRLF               = 00000033
TTY$V_SX_SKIPLF                 = 0000002D
TTY$V_SX_TABEXPAND              = 00000035
TTY$V_SX_TABRIGHT               = 0000003E
TTY$V_SX_TERMNORM               = 00000038
TTY$V_SX_TYPFUL                 = 0000002C
TTY$V_SX_WRAP                   = 0000002F
TTY$V_SX_WRITE                  = 00000007
TTY$V_SX_WRTALL                 = 00000024
TTY$WRITEDONE                   ******** X   02
TTY$W_RB_AESLEN                 = 00000028
TTY$W_RB_CPZCUR                 = 00000038
TTY$W_RB_CPZORG                 = 0000003A
TTY$W_RB_ESCTKN                 = 00000048
TTY$W_RB_LINOFF                 = 00000030
TTY$W_RB_LINREST                = 00000032
TTY$W_RB_MODE                   = 00000044
TTY$W_RB_PRMLEN                 = 00000034
TTY$W_RB_RDSTATE                = 0000002A
TTY$W_RB_TXTOFF                 = 0000003C
TTY$W_RB_TXTSIZ                 = 00000040
TTY$W_TA_INAHD                  = 0000000C
TTY$W_TA_RCLOFF                 = 0000000E
TTY$W_TA_RCLSIZ                 = 00000014
TTY$W_WB_BCNT                   = 0000002A
TTY$W_WB_STATUS                 = 00000028
TTY$XON                         ******** X   02
TT_END                          00001583 R   02
UCB$B_DEVTYPE                   = 00000041
UCB$B_TT_CRFILL                 = 000000F6
UCB$B_TT_ESC                    = 00000103
UCB$B_TT_ESC_O                  = 00000104
UCB$B_TT_FILL                   = 00000102
UCB$B_TT_LASTC                  = 000000FF
UCB$B_TT_LFFILL                 = 000000F7
UCB$B_TT_LINE                   = 000000FE
UCB$B_TT_OUTYPE                 = 0000010B
```

```
UCB$L_DEVDEPEND                 = 00000044
UCB$L_DEVDEPND2                 = 00000048
UCB$L_DUETIM                    = 0000006C
UCB$L_IRP                       = 00000058
UCB$L_SVAPTE                    = 00000078
UCB$L_TT_MULTI                  = 000000D8
UCB$L_TT_OUTADR                 = 0000011C
UCB$L_TT_RDUE                   = 000000B0
UCB$L_TT_SMLT                   = 000000E0
UCB$L_TT_TYPAHD                 = 000000E4
UCB$L_TT_WRTBUF                 = 000000D4
UCB$M_INT                       = 00000002
UCB$M_TIM                       = 00000001
UCB$Q_TT_STATE                  = 000000B8
UCB$V_INT                       = 00000001
UCB$V_TT_TIMO                   = 00000001
UCB$W_BCNT                      = 0000007E
UCB$W_BOFF                      = 0000007C
UCB$W_BUFQUO                    = 00000018
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_STS                       = 00000064
UCB$W_TT_BSPLEN                 = 00000100
UCB$W_TT_CURSOR                 = 000000FC
UCB$W_TT_MULTILEN               = 000000DC
UCB$W_TT_OUTLEN                 = 00000120
UCB$W_TT_PRTCTL                 = 00000122
UCB$W_TT_SMLTLEN                = 000000DE
UPDATE_CURSOR                     000010A9 R      02
UPDCURSOR                         00000EE6 R      02
VERIFY_ARRAY                      ******** X    02
VTAB                              000003BE R      02
W0                              = 00000000
W1                              = 00000001
WRITE_END                         00000093 R      02
WRITING                           0000004D R      02
X                               = 00000000
X0                              = 00000000
X1                              = 00000001
XON                               00000632 R      02
Z0                              = 00000003
Z1                              = 00000001
ZERO_CPZORG                       00001118 R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00001583 (5507.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.04	00:00:00.75
Command processing	117	00:00:00.35	00:00:02.68
Pass 1	880	00:00:31.65	00:01:18.64
Symbol table sort	0	00:00:03.01	00:00:05.46
Pass 2	533	00:00:08.06	00:00:16.13
Symbol table output	1	00:00:00.22	00:00:00.34
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1568	00:00:43.34	00:01:44.01

The working set limit was 2400 pages.
273339 bytes (534 pages) of virtual memory were used to buffer the intermediate code.
There were 150 pages of symbol table space allocated to hold 2533 non-local and 289 local symbols.
3477 source lines were read in Pass 1, producing 29 object records in Pass 2.
52 pages of virtual memory were used to define 49 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	20
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	29

2791 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:TTYCHARO/OBJ=OBJ$:TTYCHARO MSRC$:TTYCHARO/UPDATE=(ENH$:TTYCHARO)+EXECML$/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TTYFDT
LIS

TTYCHARO
LIS

TTYCHARI
LIS

TTYDRVDAT
LIS